INFO – 556
INFORMATION RETRIEVAL & WEB SEARCH

PROJECT REPORT

Project Title

Carnatic Music Raga Identification using Music Information Retrieval + Deep Learning

Mentored by
Dr. Xiao Hu
Associate Professor,
College of Information Science, University of Arizona

Submitted by

Student ID	*23946913*
Name	*Sundar Ram Subramanian*
Date	*12/06/2025*

TABLE OF CONTENTS

TABLE OF TABLES

TABLE OF FIGURES

PROJECT OVERVIEW

This project aims to develop a deep learning based Carnatic Music Raga Retrieval System capable of automatically identifying the raga of Carnatic music clips. It integrates Music Information Retrieval (MIR) techniques for extracting characteristic Raga features from digital audio with modern deep neural networks for Raga classification. Patented dataset Indian Art Music Raga Recognition datasets [1] of UPF has been leveraged for feature extraction and to train and evaluate models for raga classification of 8 Melakarta ragas. Python's Librosa library is used for feature extraction from audio file and a Fusion Deep Learning Model (CNN + LSTM) with SoftMax output has been built for Raga Identification and Classification.

ABOUT CARNATIC MUSIC

Carnatic music is one of the world's oldest and most deeply rooted musical traditions. [8]. Raga is the single most crucial foundational melodic framework of Indian classical music that provides the rules and notes for a Carnatic composition and shapes its expression.[9]. It contains a primordial sound (nāda), tonal system (swara), pitch (śruti), scale, and characteristic ornaments (gamakas). There are several number of Ragas in this music art form that is based on 72 Melakarta (parent) Ragas. These 72 Melakarta Ragas are formed by certain rules and constraints to the 7 tones & 5 semitones (swaras). [12]. All other Ragas called Janya Ragas are derived from these melakartha ragas.

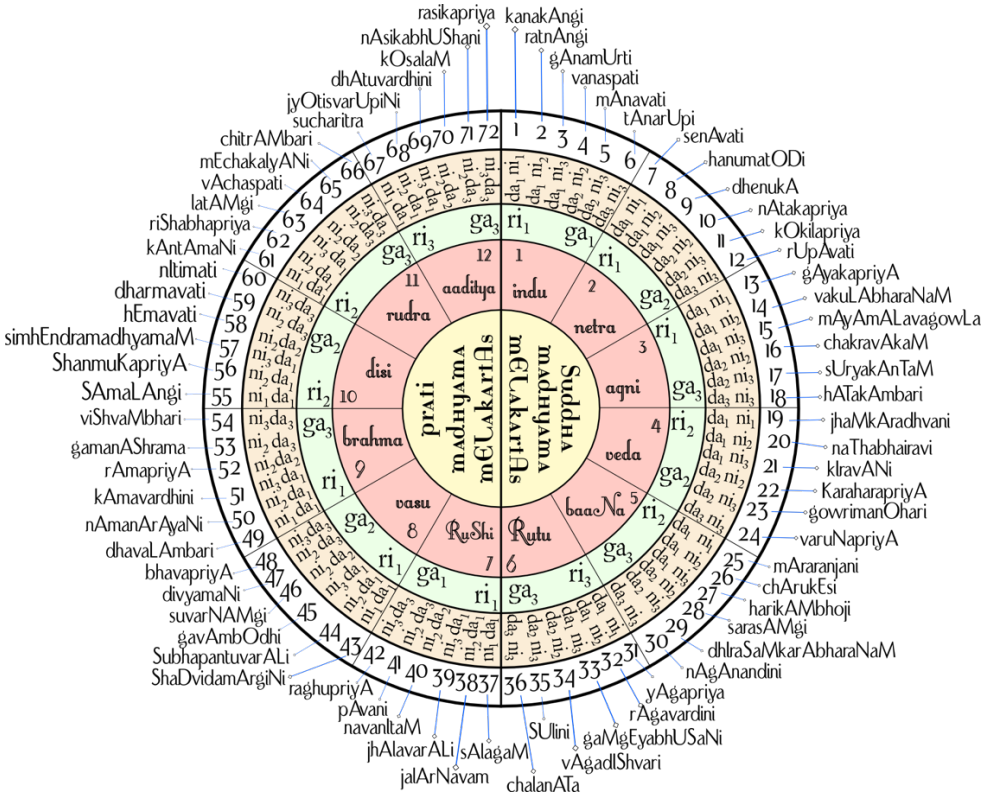

Figure 1 Melakarta Ragas (Source: Wikipedia) [13]

PROJECT MOTIVATION & PURPOSE

Research shows that music can meaningfully influence task performance, including in demanding environments such as surgical operating rooms. [11]. This becomes particularly interesting when considering Carnatic ragas, whose precise and well-structured melodic patterns are known to evoke distinct moods and emotional responses [14]. If these emotional characteristics can be reliably identified from an audio signal, it opens the possibility of automatically recognizing ragas in unseen recordings. Such recognition enables the creation of a more responsive recommendation system one capable of selecting musically similar compositions that enhance and support the user's ongoing activity. Also, learning to identify a song's raga is a core competency developed during an Indian Carnatic music education, and efforts to develop computational raga identifiers have thus gained traction [10]. As a starter, this project leverages Music Information Retrieval techniques to extract and analyse Carnatic audios for 8 melakarta ragas, and build a fusion deep learning model (CNN+LSTM) for Raga classification. A fusion of CNN & LSTM is particularly important to consider both static and temporal information that are characteristic of the Raga.

GOAL

The goal of this project is to leverage Music Information Retrieval Techniques to successfully extract Sruthi (Tonic Sa or Fundamental Frequency) from an audio, standardize the audio to common Sruthi (C3), extract characteristic Raga Features and finally design and develop a deep learning fusion model (CNN + LSTM) that can automatically identify the raga (at least 5 ragas) of an unseen Carnatic music clip.

BENEFITS

This project offers broad impact across Information retrieval, cultural preservation, and human-centered computing. Automatic raga identification enables intelligent recommendation systems that adapt music to mood, wellness, or task performance, while also providing powerful educational tools for learners, instructors, musicians and listeners. The system's ability to map ragas to emotional and cognitive states also strengthens applications in therapy, meditation, and focus enhancement, where curated sound can meaningfully influence user wellbeing. In creative domains, raga-aware models can guide AI-assisted composition tools that maintain melodic rules and integrity. Moreover, integrating automatic identification into music platforms can significantly improve metadata quality, enabling more accurate search, discovery, and organization of classical recordings. Overall, this work contributes to more inclusive AI systems that understand diverse musical traditions and lays a scalable foundation for future research and industry applications.

CORE FUNCTION & TECHNICAL APPROACH

Methodology

Figure 2 Methodology

Dataset Collection and Preparation:

The following datasets were collected & explored:

1. Carnatic Song Database, Kaggle [16] – Dropped in need to download YT videos.
2. Saraga Carnatic Music Dataset, Kaggle [17] – Dropped due to very complex structure of dataset.
3. Ragas with features in Indian classical music, Kaggle [18] – Dropped due to unavailability of audio files.
4. Carnatic Janaka Ragas, Kaggle [19] – Dropped, No relevant details except that of Raga names and swaras.
5. The Indian Art Music Raga Recognition datasets [1] – Selected, Huge dataset with audio in mp3 format and proper labelling of Ragas in hierarchical directories (Ref Fig 2).

The Indian Art Music Raga Recognition datasets [1] was shortlisted to further the project goal.

```
RagaDataset/
├── Carnatic
│   ├── audio
│   ├── features
│   └── _info_
└── Hindustani
    ├── audio
    ├── features
    └── _info_
```

Figure 3 Dataset Hierarchy

For this project, 8 Melakarta ragas were shortlisted namely; Hanumathodi, Harikambhoji, Kaamavardhini, Kalyani, Kharaharapriya, Mayamalavagowla, Shankarabharanam, and Shanmugapriya. These ragas were selected based on practical and musical considerations: the availability of shorter audio recordings under 15 minutes, which is important given the computational limits of processing long audio on a local machine; the requirement that each be a Melakarta (parent) raga to avoid the added complexity introduced by derivative Janya raga structures; and their widespread use in performance, ensuring that users can meaningfully appreciate and interpret the model's classification results.

The Ragas in the directories were encoded miscellaneously and the __*info*__ directory had details of the encodings (Refer Fig 3). Minimum of 5 songs from each of the shortlisted 8 ragas were selected based on their audio size and a separate sample dataset having only audios were created.

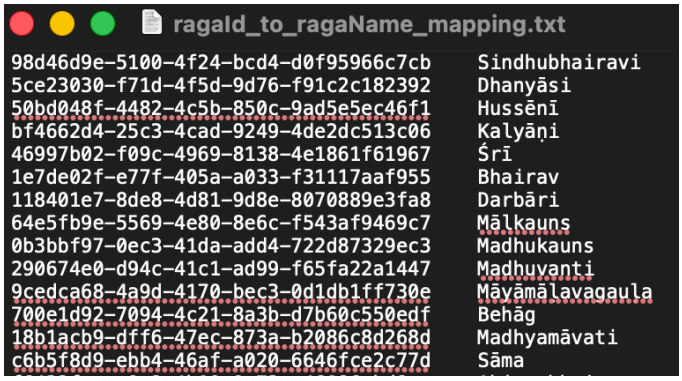

Figure 4 Raga Encodings in the Directory

The remaining blocks (Audio Preprocessing, Feature Extraction & EDA, Model Building and Model Evaluation) involves python coding and explained in detail in the upcoming sections.

Code Architecture

This project leverages python coding across Microsoft Visual Studio code and Google collab. Feature extraction was performed in Visual Studio code while EDA & Model Building was performed in Collab as the feature dataset size was very enormous for a local machine to read and run the file. Refer to Table 1 for the list of python libraries and their purpose in this project.

Library	Purpose
librosa	Digital Signal Processing for Feature Extraction
tensorflow	Deep Fusion Neural Network Model Building, Training & Evaluation
keras	
numpy	for handling numeric arrays of embeddings
pandas	for structured data storage & data manipulation in dataframes
scipy	for statistical operations of numerical arrays
scikit-learn	For Encoding, Dataset splitting & Model Evaluations
matplotlib	for Visualization
umap-learn	
openpyxl	for writing csv
pyarrow	for pickle & parquet fikle creation

Table 1 Python Libraries used

All the blocks except data collection & preparation are built in to 3 different pipelines as in Figure 4. During the course of the project, I realised that Audio Processing & Feature extraction had to happen in a jumbled fashion to extract relevant features. The Feature Extraction was divided in to two parts (based on

domain knowledge) as (i) Shruti Extraction & Standardization (Tonic Pitch – Fundamental Frequency) & (ii) Identification of Raga features & extraction. Both of these parts are built as first two sequential pipelines.

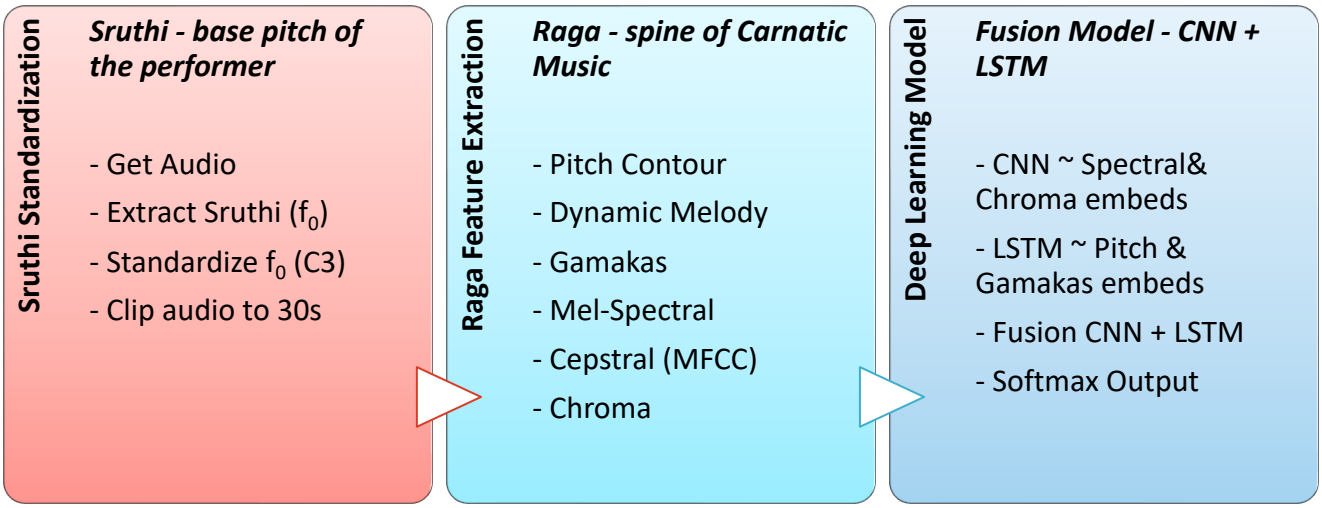

Figure 5 Code Architecture - Pipelines

Shruti Extraction & Standardization pipeline:

Shruti refers to the foundational pitch or frequency chose by the performer. [15]. All the subsequent notes get the frequency position depending on the starting frequency. This key point is that Indian musical notes are all relative based on the starting position. This foundational musical note is called the Shajama (or Shadja) which is sung as 'Sa' in short. Only after establishing the position of this 'Sa' can we move on to exploring the rest of the musical notes. In western music terminology, this foundational pitch is called the *Root note or Tonic* and can be compared to the note 'C'. librosa library uses piptrack which in turn leverages Short-Time Fourier Transform (STFT) magnitude to detect local peaks per STFT time frame. These pitch candidates are further processed for pitch class histogram & Peak Detection that gives the final Shruti of the original audio. Once this Fundamental Frequency (in Hz) is identified, librosa is again used to convert the fundamental frequency to a MIDI (Musical Instrument Digital Interface) number, and in turn map the number to the specific note (swara) to which note it corresponds to. This note represents the shruti of the artist.

Each artist have their own Shruti and hence it becomes important to standardize the Shruti before extracting the features of the corresponding Raga. A standard frequency corresponding to C3 (130.8128 Hz) is selected and all the audio features are converted to this standard frequency using the pitch_shift function of librosa.effects.

Post Shruti standardization, each of the shruti standardized audio has been split in to clips of 30 seconds each as an effort to increase the number of samples from which the Fusion Deep Learning model can learn from. This step precedes the Raga Feature Extraction as it is essential to extract the Raga features for each of the

sample while Shruti remains the same for all the samples which is C3 (130.8128 Hz). Refer Figure 5 for the entire pipeline.

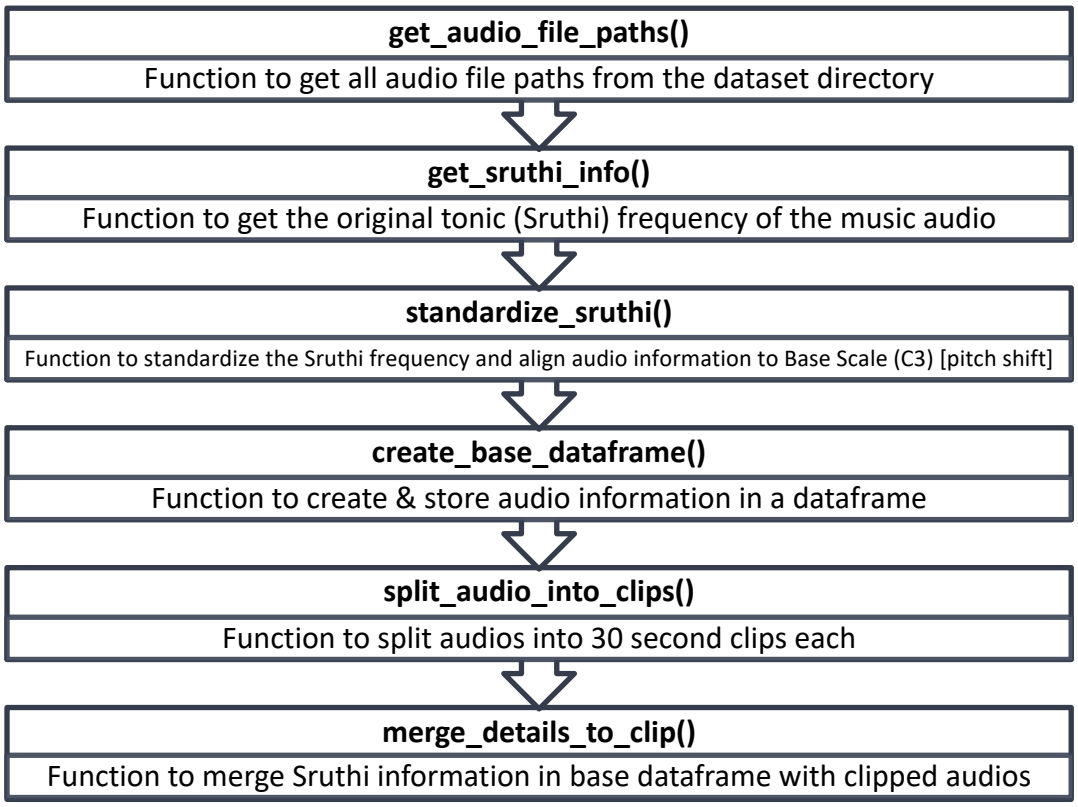

Figure 6 Sruthi Identification & Standardization Pipeline

Raga Features Extraction pipeline:

Once all audio clips are shruti standardized to a common tonic (C3), the next pipeline focuses on extracting musically meaningful features that characterize each Raga. Carnatic ragas are defined not only by their swara set but also by the way these swaras are approached, ornamented, emphasized, and sequenced across time. Therefore, the feature extraction pipeline captures pitch based melodic movement, interval sequences, gamaka behavior, and spectral timbre cues from each standardized audio segment. The pipeline is structured as a series of modular steps, where each block extracts one family of descriptors and appends them to the growing feature dataframe. Refer Figure 6.

The pipeline begins with pitch contour extraction, which forms the foundation for all melody-driven descriptors. Using the `librosa.pyin` algorithm, each audio clip is converted into a frame-wise fundamental frequency (f_0) sequence, expressed both in Hertz and in cents relative to the standardized tonic (C3). This representation enables the model to capture raga specific melodic shapes, characteristic phrases, gamaka curves, and microtonal movements. The pitch contour is also converted into a pitch-class sequence (mod 1200), which provides octave-invariant information regarding the distribution of swaras used within the clip.

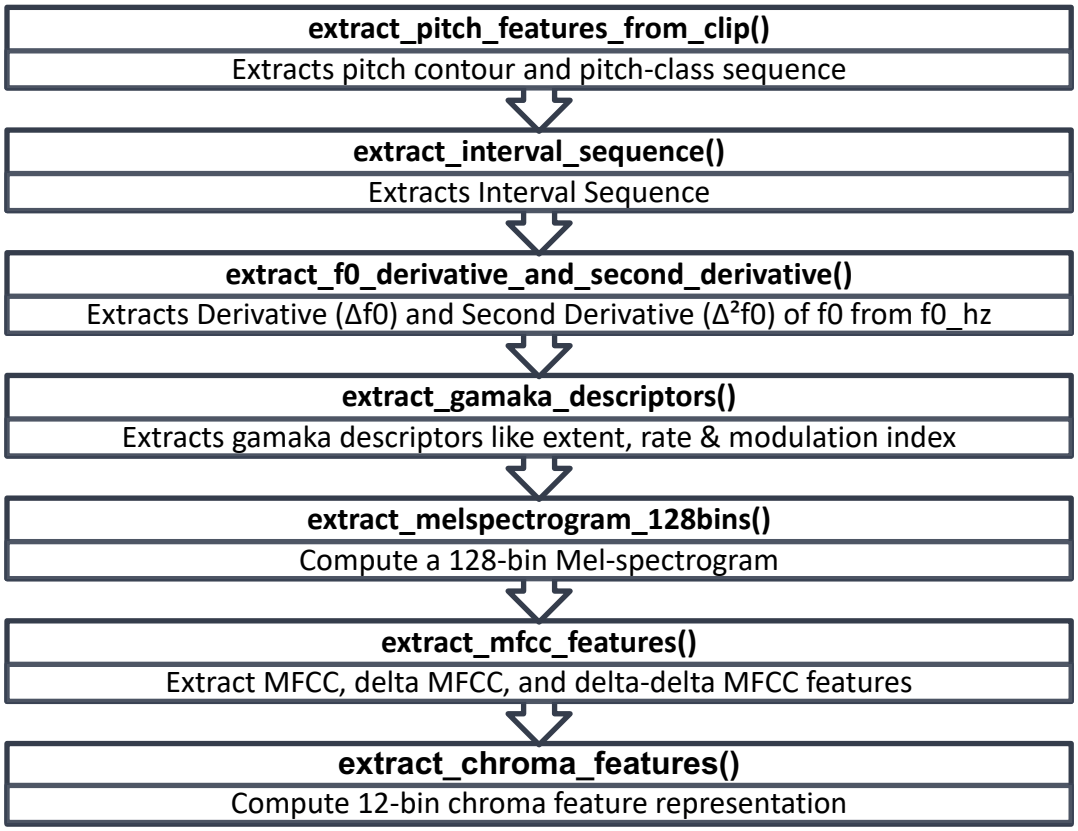

Figure 7 Raga Feature Extraction Pipeline

Building on the f_o contour, the next block computes the interval sequence, defined as successive differences of f_o in the cent domain. These intervals encode the ascending–descending behavior (arohana–avarohana), relative swara transitions, and the spacing between melodic steps, allowing the pipeline to represent both scale structure and phrase level melodic evolution. Following this, the first and second derivatives of the f_o contour are extracted in the Hertz domain. While interval sequences measure musical steps, the derivatives characterize rate and smoothness of pitch movement, directly capturing the dynamics of gamakas, bends (jaarus), kampita oscillations, and vibrato-like modulations.

Using these derivative signals, the next stage computes Gamaka Descriptors, a set of MIR informed features that quantify three core aspects of Carnatic ornamentation (Gamakas):

1. Gamaka Extent, a measure of how widely the pitch oscillates around its centre, computed as the local standard deviation of f_o in cents
2. Gamaka Rate, estimated through the zero-crossings of the f_o derivative, which approximates the speed of oscillation; and
3. Modulation Index, defined as the ratio of extent to rate, capturing the overall strength and prominence of gamaka behavior within each window.

These descriptors together allow the system to encode expressive features that strongly differentiate ragas such as Bhairavi (wide oscillations) from Kalyani (more controlled, mild modulation).

Beyond melodic descriptors, the pipeline extracts spectral features that capture timbre, harmonic structure, and articulation. First, a 128-bin Mel-Spectrogram is computed for every clip, producing a time-frequency representation that reflects the vocal formants, harmonic envelope, and broad spectral energy distribution. This is followed by extraction of MFCCs, along with their first- and second-order deltas, which provide detailed information about vocal tract shaping, articulation, and timbral texture attributes that often correlate with raga identity, especially in alapana-style singing.

Finally, the pipeline extracts 12-bin Chroma Features, which summarize the energy distribution across the 12 pitch classes. Chroma descriptors effectively capture swara usage patterns, highlighting which notes are emphasized, how often specific swaras are visited, and the overall pitch-class profile characteristic of each raga.

All extracted features pitch contour, pitch-class sequence, interval sequence, f_o derivatives, gamaka descriptors, Mel-spectrograms, MFCCs, and chroma vectors are merged into a unified dataframe, forming a rich multi-dimensional representation of each audio sample. This comprehensive feature set is saved as a pickle file. This file serves as the input for the Deep learning CNN + LSTM Fusion Model enabling it to learn both time frequency structure and melodic ornamental patterns.

Exploratory Data Analysis

The following analysis was performed to understand standardization of Sruthi and how certain features well differentiate ragas. Following visualizations were explored to understand the aforementioned aspects.

1. Distribution of f0 in Hz across Ragas

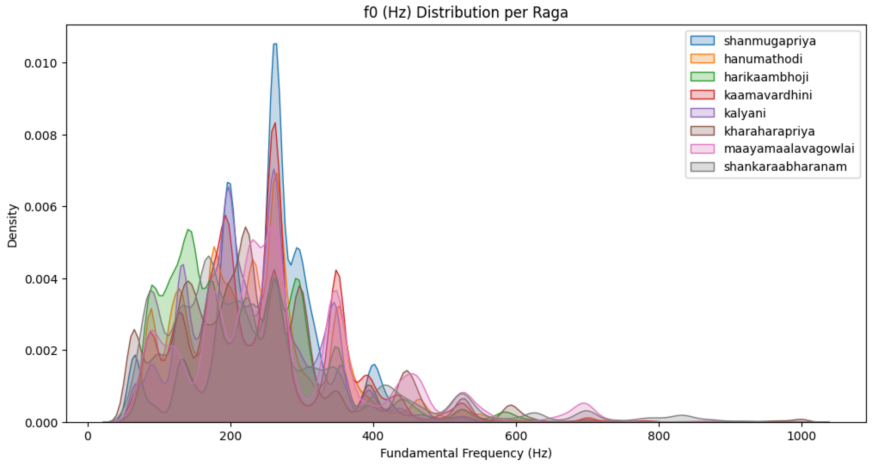

Figure 8 Distribution of f0 in Hz across Ragas

It can be seen that the Sruthi across audio clips are standardized and all ragas sit in a common pitch range. Each raga has unique pitch density "fingerprints". Even with massive overlap, subtle peaks differ.

2. Pitch Class Distribution

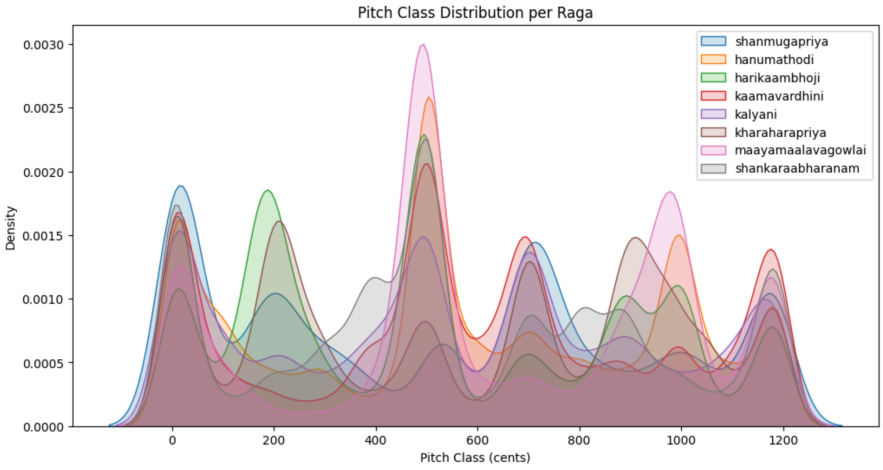

Figure 9 Pitch Class Distribution

All ragas share strong peaks at Sa (0 cents), Pa (500 cents), and upper Sa (1200 cents) due to tonic fixation and common panchama. The discriminative power lies in Ri, Ga, Ma, Dha, Ni peaks. Each raga shows different prominence, sharpness, and peak shifting. Ragams are separable based on pitch-class distributions despite sharing the same tonic. Ga, Ma, Ni positions and widths offer maximum discriminative power.

3. Median Pitch Contour Shape

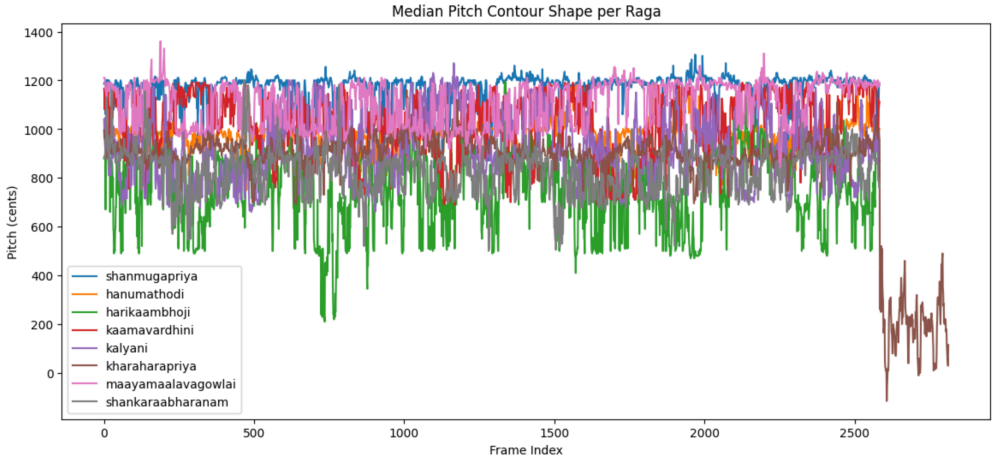

Figure 10 Median Pitch Contour Shape

This Median Pitch Contour captures the average melodic fingerprint of each raga across all clips. Shanmukhapriya shows the most stable, plateau-like contour. Harikambhoji shows strong downward fluctuations — a key identifying trait. Maayamaalavagowla displays large oscillations representing intense kampita. Shankarabharanam shows mid-range stability with lower sthayi dips toward clip endings.

4. Mean Chroma Profile

From figure 10, we can see each line represents average energy across swaras. This indicates how strongly each swara is emphasized in a raga. Different ragas show distinct swara emphasis patterns. Sa and Pa

remain stable anchors across all ragas. Discriminative differences appear mainly in other swaras. Gamaka-heavy ragas show more energy variation, while scale-based ragas show smooth energy curves.

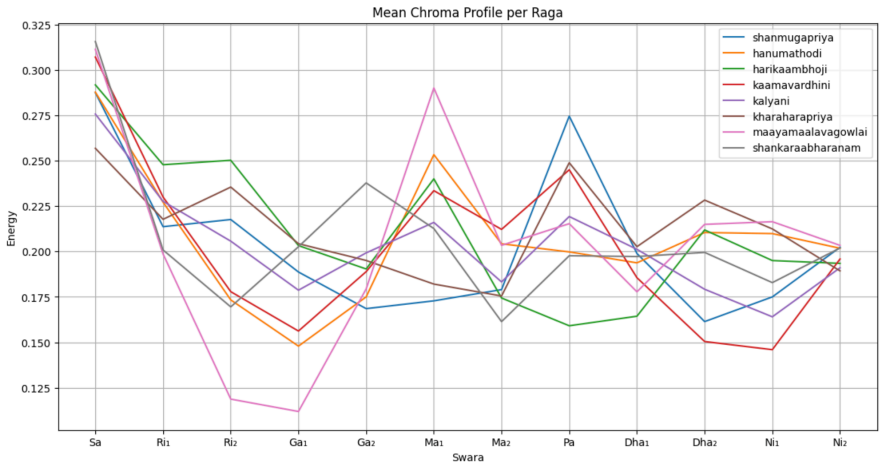

Figure 11 Mean Chroma Profile

5. Mel Spectrogram Heatmap

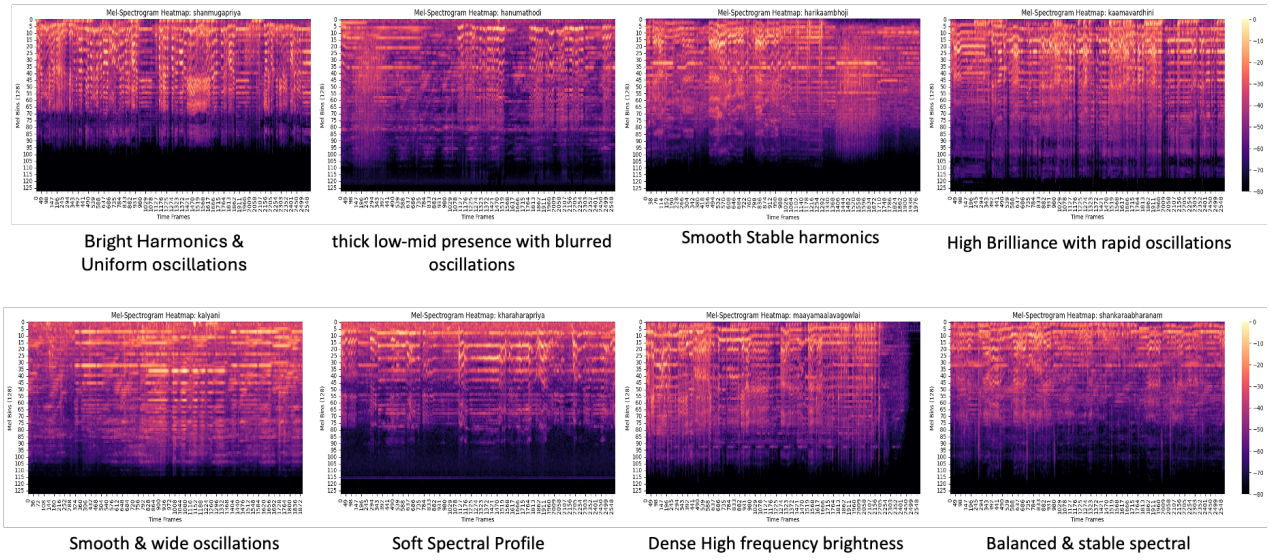

Figure 12 Mel Spectrogram Heatmap across Ragas

A Mel-Spectrogram shows how the energy of different frequency bands evolves over time. We could observe the following features in these charts on Figure 11. The Mel-Spectrogram visualizations of the eight selected Melakarta ragas reveal clear differences in their time–frequency structure, capturing the spectral characteristics that distinguish each raga. The heatmaps show how energy is distributed across mel frequency bands over time, making visible the unique melodic contours, gamaka behavior, and swara emphasis inherent to each raga. Ragas with heavy oscillatory ornamentation, such as Hanumatodi and Kaamavardhini, exhibit dense frequency modulations and sweeping patterns, while smoother ragas like Kalyani and Shankarabharanam display more stable harmonic bands.

6. Raga Raga Correlation

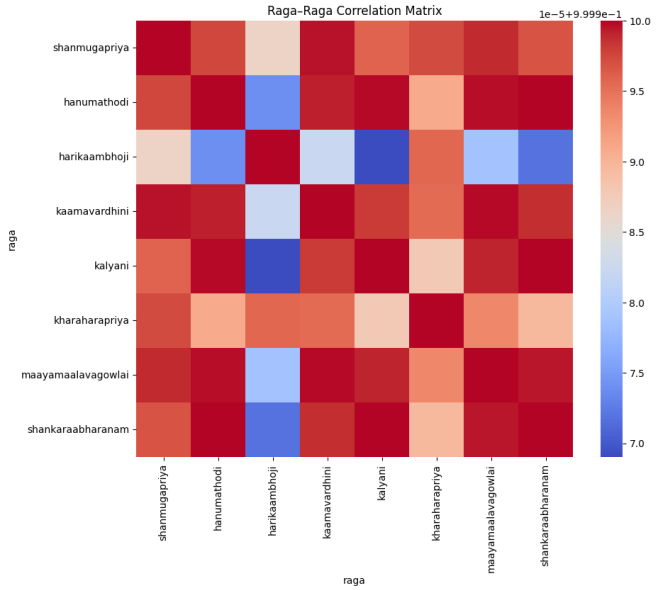

Figure 13 Raga Raga Correlation

From Figure 12, it is evident that most of the Ragas are correlated and hence a lot of bright and dark reds. This indicates the major challenge in Raga Identification.

Deep Learning Fusion Model Pipeline

After extracting a comprehensive set of melodic, gamaka-related, spectral descriptors from each 30-second shruti-standardized audio segment and EDA, a Deep Learning Fusion model was designed to learn the multidimensional structure of Carnatic ragas. The classification task requires a model that can simultaneously understand time–frequency patterns, ornamentation dynamics, and long-range melodic trajectories. To meet these goals, a Fusion Deep Learning Architecture that integrates a Convolutional Neural Network (CNN) branch and an LSTM-based sequential branch, enabling it to jointly learn spectral–timbre information and temporal–melodic behavior.

The CNN branch processes the Mel-Spectrogram, which is reshaped and passed through a series of convolutional and pooling layers that extract local time–frequency patterns, such as harmonic envelopes, formant structures, and energy bands associated with characteristic raga timbres. These convolutional layers produce an embedding that summarizes the spectral evolution of the audio clip, serving as a compact representation of its acoustic texture. The CNN model summary is as displayed below in Table 2.

The CNN branch of the fusion architecture processes the 128-bin Mel-Spectrogram input through a sequence of convolutional, normalization, and pooling operations. The network begins with an input tensor of shape (153, 2813, 1), representing the Mel-Spectrogram frames, and applies three successive Conv2D blocks with increasing filter depths (32 → 64 → 128). Each convolutional layer is followed by Batch Normalization to

stabilize training and maintain numerical consistency, and Max Pooling layers that progressively reduce the time–frequency resolution while preserving salient spectral features. This hierarchical down sampling allows the model to capture localized harmonic structures, formant patterns, and time–frequency textures unique to each raga. After these convolutional stages, the representation is flattened into a high-dimensional vector (~850k elements) and passed through a dropout layer to reduce overfitting. Finally, a Dense layer compresses the flattened output into a compact 256-dimensional CNN embedding, which serves as the spectral feature representation for the fusion model. Overall, this branch contains approximately 218 million trainable parameters.

Model: "functional"

Layer (type)	Output Shape	Param #
cnn_input (InputLayer)	(None, 153, 2813, 1)	0
conv2d (Conv2D)	(None, 153, 2813, 32)	320
batch_normalization (BatchNormalization)	(None, 153, 2813, 32)	128
max_pooling2d (MaxPooling2D)	(None, 76, 1406, 32)	0
conv2d_1 (Conv2D)	(None, 76, 1406, 64)	18,496
batch_normalization_1 (BatchNormalization)	(None, 76, 1406, 64)	256
max_pooling2d_1 (MaxPooling2D)	(None, 38, 703, 64)	0
conv2d_2 (Conv2D)	(None, 38, 703, 128)	73,856
batch_normalization_2 (BatchNormalization)	(None, 38, 703, 128)	512
max_pooling2d_2 (MaxPooling2D)	(None, 19, 351, 128)	0
flatten (Flatten)	(None, 853632)	0
dropout (Dropout)	(None, 853632)	0
cnn_embedding (Dense)	(None, 256)	218,530,048

Total params: 218,623,616 (833.98 MB)
Trainable params: 218,623,168 (833.98 MB)
Non-trainable params: 448 (1.75 KB)

Table 2 CNN Model Summary

In parallel, the LSTM branch processes pitch-driven sequential features, such as f_0 contour, its derivatives, interval sequences, gamaka descriptors, and chroma vectors. These sequential inputs are passed through stacked LSTM layers producing an embedding that summarizes the long-duration melodic dependencies, gamaka trajectories, swara landing patterns, and phrase-level raga structures. The CNN model summary is as displayed below in Table 3.

The LSTM branch receives an input tensor of shape (2810, 7), representing frame-wise melodic descriptors such as f_0 contour, interval sequence, derivatives, and chroma features. The core of this branch is a Bidirectional LSTM layer with 256 output units, enabling the model to learn temporal patterns both forward and backward in time. The BiLSTM transforms the raw sequence into a rich temporal embedding while preserving long-range dependencies and microtonal variations. A dropout layer follows to prevent overfitting and improve generalization. Subsequently, a Dense layer reduces the representation to a 128-dimensional LSTM embedding, providing a compact summary of the melodic and temporal structure of each audio clip. With approximately

172,160 trainable parameters, this branch is computationally lightweight compared to the CNN branch, yet crucial for modeling the expressive and sequential aspects of ragas that cannot be captured by spectral features alone.

Model: "functional_1"

Layer (type)	Output Shape	Param #
lstm_input (InputLayer)	(None, 2810, 7)	0
bilstm (Bidirectional)	(None, 256)	139,264
lstm_dropout (Dropout)	(None, 256)	0
lstm_dense (Dense)	(None, 128)	32,896
dropout_1 (Dropout)	(None, 128)	0

Total params: 172,160 (672.50 KB)
Trainable params: 172,160 (672.50 KB)
Non-trainable params: 0 (0.00 B)

Table 3 LSTM Model summary

The outputs of both branches are concatenated in a Fusion Layer. The fusion stage integrates the CNN and LSTM branches by concatenating their learned embeddings into a unified representational space producing a unified embedding that captures both spectral and melodic identity. This fused embedding is then passed through a softmax based classification layer trained using the Adam optimizer with categorical cross-entropy loss. The fusion model architecture is shown in Figure 13.

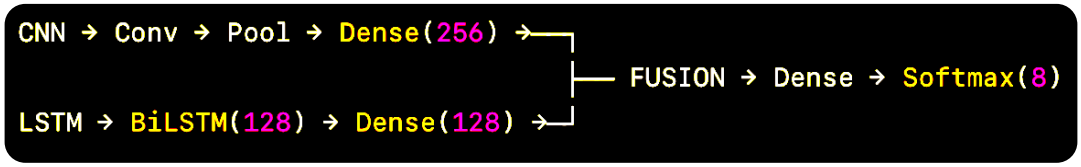

Figure 14 Fusion Model Architecture

The 256-dimensional CNN embedding, which captures spectral patterns and timbral characteristics, is combined with the 128-dimensional LSTM embedding, which encodes melodic transitions, gamaka dynamics, and temporal structure. This results in a 384-dimensional fused feature vector that jointly represents both time–frequency and sequential aspects of the input audio. Following concatenation, the fusion vector passes through a dropout layer to reduce overfitting and a Dense layer that compresses it into a 256-dimensional fusion embedding. Finally, a softmax classification layer maps this fused representation to the eight target ragas. This fusion design ensures that the classifier benefits from both spectral discriminability and melodic expressiveness, resulting in a more robust and musically informed raga recognition system. Table 4 provides the summary of entire Fusion model summary.

Model: "functional_2"

Layer (type)	Output Shape	Param #	Connected to
cnn_input (InputLayer)	(None, 153, 2813, 1)	0	–
conv2d (Conv2D)	(None, 153, 2813, 32)	320	cnn_input[0][0]
batch_normalization (BatchNormalizatio…	(None, 153, 2813, 32)	128	conv2d[0][0]
max_pooling2d (MaxPooling2D)	(None, 76, 1406, 32)	0	batch_normalizat…
conv2d_1 (Conv2D)	(None, 76, 1406, 64)	18,496	max_pooling2d[0]…
batch_normalizatio… (BatchNormalizatio…	(None, 76, 1406, 64)	256	conv2d_1[0][0]
max_pooling2d_1 (MaxPooling2D)	(None, 38, 703, 64)	0	batch_normalizat…
conv2d_2 (Conv2D)	(None, 38, 703, 128)	73,856	max_pooling2d_1[…
batch_normalizatio… (BatchNormalizatio…	(None, 38, 703, 128)	512	conv2d_2[0][0]
lstm_input (InputLayer)	(None, 2810, 7)	0	–
max_pooling2d_2 (MaxPooling2D)	(None, 19, 351, 128)	0	batch_normalizat…
bilstm (Bidirectional)	(None, 256)	139,264	lstm_input[0][0]
flatten (Flatten)	(None, 853632)	0	max_pooling2d_2[…
lstm_dropout (Dropout)	(None, 256)	0	bilstm[0][0]
dropout (Dropout)	(None, 853632)	0	flatten[0][0]
lstm_dense (Dense)	(None, 128)	32,896	lstm_dropout[0][…
cnn_embedding (Dense)	(None, 256)	218,530,0…	dropout[0][0]
dropout_1 (Dropout)	(None, 128)	0	lstm_dense[0][0]
fusion_concat (Concatenate)	(None, 384)	0	cnn_embedding[0]… dropout_1[0][0]
fusion_dropout (Dropout)	(None, 384)	0	fusion_concat[0]…
fusion_dense (Dense)	(None, 256)	98,560	fusion_dropout[0…
output (Dense)	(None, 8)	2,056	fusion_dense[0][…

Total params: 218,896,392 (835.02 MB)
Trainable params: 218,895,944 (835.02 MB)
Non-trainable params: 448 (1.75 KB)

Table 4 Fusion Model Summary

Subsequently in the pipeline, the dataset is split into training, validation, and test sets using stratified sampling to preserve raga balance. Labels are one-hot encoded, and the model is trained with early stopping, checkpointing, and learning-rate scheduling to ensure stability and prevent overfitting. Throughout training, accuracy and loss curves are monitored to validate convergence and observe generalization behavior.

The training and validation accuracy curves in figure 14 show a steady upward trend across epochs, indicating that the fusion model progressively learns the spectral and melodic patterns that distinguish the eight

ragas. Validation accuracy closely tracks the training accuracy, demonstrating good generalization with minimal overfitting. By around 35–40 epochs, the model stabilizes in the 65–75% accuracy range.

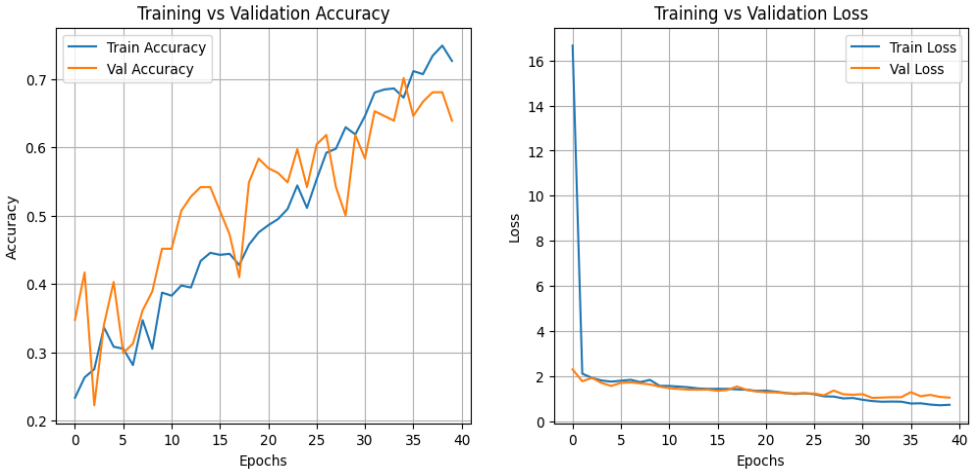

Figure 15 Training Vs Validation (Accuracy & Loss)

The loss curves in figure 14 display a rapid initial drop, especially in the first few epochs, where the model adjusts its weights to capture core raga-level features. After this sharp decline, both training and validation loss continue to decrease gradually and remain closely aligned throughout training. This indicates that the model is not overfitting to the training set. The smooth convergence of both loss and accuracy curves suggests that the learning rate, regularization strategy (dropout), and network architecture (CNN + LSTM fusion) are well-balanced for the task. Overall, these plots confirm that the model is learning effectively and maintaining stable performance across both training and unseen validation samples.

Finally, the pipeline evaluates model performance using accuracy, precision, recall, F1-score, and confusion matrices. The final evaluation on the held-out test dataset yields a test accuracy of **64.58%** and a test loss of **1.04**. This accuracy, though not very good is consistent with the training accuracy, indicating that model doesn't overfit.

CLASSIFICATION REPORT:

	precision	recall	f1-score	support
hanumathodi	0.70	0.79	0.74	29
harikaambhoji	0.48	0.67	0.56	15
kaamavardhini	0.61	0.61	0.61	18
kalyani	0.00	0.00	0.00	11
kharaharapriya	0.87	0.90	0.89	30
maayamaalavagowlai	0.78	0.50	0.61	14
shankaraabharanam	0.52	0.75	0.61	20
shanmugapriya	0.00	0.00	0.00	7

Table 5 Classification Report

The classification report in Table 4 provides a detailed breakdown of model performance across the eight Melakarta ragas. The model achieves strong precision and recall for Kharaharapriya (F1 = 0.89) and Hanumatodi

(F1 = 0.74) indicating that these ragas possess distinctive melodic–spectral signatures that the model learns reliably. Ragas such as Kaamavardhini, Mayamalavagowla, and Shankarabharanam also show moderate F1-scores (0.56–0.61), suggesting consistent but not perfect discrimination, likely due to overlaps in gamaka patterns and shared swaras with other ragas. However, the model struggles to correctly classify Kalyani and Shanmugapriya, both showing precision and recall values of zero.

Beyond classification, the pipeline also constructs an Embedding Model, a variant of the trained fusion network truncated before the softmax layer. This embedding model transforms each audio clip into a fixed-length vector that encodes its raga-specific melodic and spectral properties. To visualize the structure of ragas in this learned latent space, the pipeline applies t-distributed stochastic neighbor embedding for dimensionality reduction, projecting the high-dimensional embeddings into two dimensions. From the t-SNE plot of Raga Embeddings in figure 15, it is evident why the classification report shows extremely good results for certain ragas like hanumathodi & Kharaharapriya, while performing poorly for Kalyani and shanmugapriya. The former ragas form a well separated clusters indicating the good model performance, while the latter has shared embedding space with other ragas, causing poor precision & recall values.

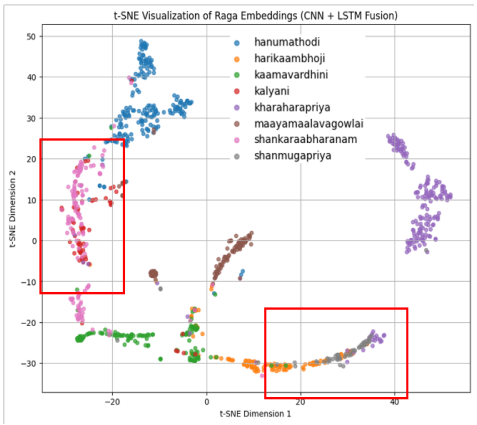

Figure 16 t-SNE plot of Raga embeddings

Class	Frequency
hanumathodi	194
harikaambhoji	102
kaamavardhini	125
kalyani	69
kharaharapriya	195
maayamaalavagowlai	93
shankarabharanam	130
shanmugapriya	49

Table 6 Class Imbalance

However, for understanding why the other ragas in the shared space were not affected, the class imbalance was measured and is evident from table 5 that, the classes are very poorly balanced, with Kalyani and shanmugapriya having only 69 and 49 samples only respectively. Therefore, they have a very low support and further contribute to reduced performance, as deep learning models generally require a balanced number of examples to learn discriminative boundaries effectively.

CHALLENGES & MITIGATION

Challenge	Description	Impact	Mitigation method
Tonic detection errors	Each singer performs in a unique pitch range. Since raga recognition depends on relative intervals, incorrect tonic detection can misalign pitch-based features and confuse the model	Misaligned features	Used established tonic detection algorithm using piptrack & pitch class peak + standardization
Complex gamakas	Carnatic vocals often include intricate pitch modulations, slides, and oscillations within a single swara. These rapid transitions make pitch tracking difficult and can cause noisy contours or misinterpretations of notes	Noisy pitch contours	High-resolution pitch extraction, smoothing (median filtering), feature fusion of standardized audio clips
Overfitting	A small dataset (few ragas, few clips per raga) can cause overfitting, especially with DNN	Low test accuracy	Regularization (dropout), CNN + LSTM Fusion modelling
Computational load	Extracting high-resolution MIR features (Mel-spectrograms, pitch contours) from large audio files is computationally heavy	Slow processing	Clip segmentation, GPU use in Collab
MIR learning curve	Much of the project involved understanding how features reflect musical meaning — how pitch contours, chroma, and timbral features relate to raga structure.	Slower progress	Visualization and iterative refinement

UNMITIGATED CHALLENGES

Challenge	Description	Impact	Reason for non-mitigation
Limited and Imbalanced Dataset	The number of vocal recordings per raga varied greatly. Some ragas have abundant data while others have very few examples.	Poor Generalization	Limited sampling due to low compute power & limited time of project
Overlapping ragas	Several Carnatic ragas share same note sets (e.g., Shankarabaranam and Kalyani) but differ in phrasing, note transitions, and gamakas	Misclassification	Limited knowledge. Need to research and understand more characteristic features

FUTURE WORK

As the model's performance strongly reflects the limitations of the current dataset, a natural direction for future work is to broaden the corpus with more diverse and balanced audio recordings, while carefully looking on opensource available infrastructure requirements. Increasing the representation of under-sampled ragas would help correct the class imbalance that significantly affects classification accuracy, while also offering the opportunity to analyze how induced emotions shape the digital signal characteristics of each raga. A richer dataset would support deeper investigations into user-centric acoustic features particularly those that capture expressive nuances such as microtonal oscillations, timbre shifts, and emotional intent enabling more discriminative and interpretable raga embeddings. Understanding which musical cues listeners rely on, and how these cues manifest in the signal domain, remains an open and valuable challenge.

Beyond these, the project naturally extends into broader exploratory and application-driven avenues. With the rapid evolution of audio transformers and multimodal architectures, there is strong potential to experiment with more powerful sequence models capable of learning highly abstract musical representations. Such advances could serve as the backbone for building intelligent interfaces, including a Shazam-like raga identification system that operates in real time and adapts recommendations based on context and emotional resonance. Given growing evidence that music can influence cognitive performance even in demanding environments [11], the ability to automatically detect raga and mood opens pathways for personalized music curation, educational support tools for learners, and richer computational understanding of Carnatic music tradition.

REFERENCES

[1] Serrà, J., Ganguli, K. K., Sentürk, S., Serra, X., & Gulati, S. (2016). Indian Art Music Raga Recognition Dataset (audio) (1.0) [Data set]. Zenodo. https://doi.org/10.5281/zenodo.7278511

[2] Sridhar, Rajeswari & Geetha, T.V. (2009). Raga Identification of Carnatic music for music Information Retrieval. SHORT PAPER International Journal of Recent Trends in Engineering. 1. Research Gate

[3] Shah, Devansh & Jagtap, Nikhil & Talekar, Prathmesh & Gawande, Kiran. (2021). Raga Recognition in Indian Classical Music Using Deep Learning. 10.1007/978-3-030-72914-1_17. Research Gate

[4] Ravikoti, Sridhar. (2020, September). Identifying Ragas in Carnatic Music with Machine Learning LinkedIn

[5] Great Learning Snippets. (2020, July). Deep Learning Based Raga Classification in Carnatic Music Medium

[6] Srinivasan, Shriya. (2021, August). Carnatic Raga Recognition Medium

[7] Divan, Shreyas. (2021, November). Raga identification using ML and DL Medium

[8] Joseph, R., & Vinod, S. (2017). Carnatic Raga Recognition. *International Journal of Science and Technology. IJST*

[9] Koduri, G.K., Gulati, S., et al. (2011). A Survey of Raaga Recognition Techniques and Improvements to the State-of-the-Art. *Sound and Music Computing.* ResearchGate

[10] Kreiman, G., & Narayanan, H. (2024). Classifying Ragams in Carnatic Music with Machine Learning Models- A Shazam for South Indian Classical Music. *Harvard Kreiman Lab.* Harvard Kreiman Lab

[11] Boghdady, M.E., Ewalds-Kvist, B,M. (2020). The influence of music on the surgical task performance: A systematic review. *International Journal of Surgery*. https://doi.org/10.1016/j.ijsu.2019.11.012

[12] Carnatic music. (2025, November 23). In Wikipedia. Wikipedia

[13] Melakarta. (2024, November 18). In Wikipedia. Wikipedia

[14] Carnatic Raga. (2025, July 20). In Wikipedia. Wikipedia

[15] Mallela, T. (2016, March 14). Introduction to Shruti – The Foundational Pitch. SUNADAM Sunadam

[16] Sanjay Natesan. (2025). Carnatic Song Database [Data set]. Kaggle. doi.org/10.34740/KAGGLE/DSV/10622900

[17] Desolationofsmaug. (2023). Saraga Carnatic Music Dataset [Data set]. Kaggle. Kaggle

[18] Sanjana satish68l. (2023). Ragas with features in Indian classical music [Data set]. Kaggle. Kaggle

[19] Pradeep Miriyala. (2021). Carnatic Janaka Ragas [Data set]. Kaggle. doi.org/10.34740/KAGGLE/DS/1446054