Raga Recognition in Indian Classical Music Using Deep Learning

Devansh P. Shah(✉), Nikhil M. Jagtap, Prathmesh T. Talekar, and Kiran Gawande

Department of Computer Engineering, Sardar Patel Institute of Technology, Mumbai, India
{devansh.shah,nikhil.jagtap,prathmesh.talekar,kiran_gawande}@spit.ac.in

Abstract. Raga is central to Indian Classical Music in both Carnatic Music as well as Hindustani Music. The benefits of identifying raga from audio are related but not limited to the fields of Music Information Retrieval, content-based filtering, teaching-learning and so on. A deep learning and signal processing based approach is presented in this work in order to recognise the raga from the audio source using raw spectrograms. The proposed preprocessing steps and models achieved 98.98% testing accuracy on a subset of 10 ragas in CompMusic dataset. A thorough study on the effect of various hyperparameters, sound source separation and silent part removal is carried out and is listed with results and findings of the same. A discussion on the predictions and behavior of deep learning models on audio apart from the dataset is also carried out. This new approach yields promising results and gives real time predictions from raw audio or recordings.

Keywords: Raga recognition · Indian Classical Music · Music Information Retrieval · Deep learning

1 Introduction

Raga is a very important aspect of Indian Classical Music (ICM). Formally, raga is a melodic framework for improvising melodic mode in music. Compositions and performances of ICM heavily rely on these ragas. In short, ragas can be termed as the grammar of Indian Classical Music. The concept of ragas is fundamental in both Hindustani as well as Carnatic Classical Music. A raga gets its identity from the *svaras* used in it. The seven svaras themselves have their own variations. The set of these svaras in unique sequence form a raga's unique property. However, some ragas can have the same set of svaras and still be different ragas depending on prominent *svaras* and the overall mood created by the characteristic phrases of the raga. For example, raga Jaunpuri and raga Darbari have the same set of notes. But raga Darbari can be distinguished from raga Jaunpuri by the characteristic oscillation around the *komal Ga* note. This oscillation is not present in raga Jaunpuri. ICM practitioners introduce their own

J. Romero et al. (Eds.): EvoMUSART 2021, LNCS 12693, pp. 248–263, 2021.
https://doi.org/10.1007/978-3-030-72914-1_17

variations in a raga to enhance melodic performance. This makes identifying raga not only interesting but a challenging task. Moreover, in Indian Classical Music, svaras are relative to each other and don't have fixed pitch value which is the case in Western Music. A performer chooses his/her base pitch of svara *Sa* and all the other svaras are relative to that. This pitch is called the Tonic Pitch of the performer. The most significant svara in a raga is termed as vadi and the second most significant svara is termed as samvadi. In a raga, svaras ascend and descend which are called arohana and avrohana respectively.

There have been various approaches to recognise the raga from music. Early approaches were based on the simple rule base heuristic, pitch tracking and hidden Markov models. The CompMusic dataset which is used in this work as well was published by Gulati et al. [6,7]. It made the data driven approach possible in this domain. These works are detailed further in the section of Literature Survey.

The intense variation of relative pitches, variation in tempos, arohana and avrohana, the tonic pitch of the performer along with his/her own style of performing particular raga makes raga identification an even more complex task. Recognising raga of a recording or performance has its own benefits. It can be used for Music Information Retrieval (MIR) of ICM, music recommendations, learning and practicing ragas, composing and many more. Moreover, it can further enhance the music experience of appreciators of music. The real time feedback from model can be used for teaching-learning, compositions and performances.

The proposed work is based on the thorough understanding of the ragas of the author Devansh Shah and his decade long experience in the field of Indian Classical Music. The need of Raga Recognition, knowledge of Ragas and the understanding of deep learning frameworks have motivated this research and experimentation. A deep learning model of hybrid architecture and preprocessing pipeline is proposed in this work which can identify raga in real time. Through this work, the hypotheses of authors that the models will perform better with sound source separation of audio, models will have less confident predictions in the faster tempo parts of audio and removal of the silent part of audio will improve performance are also validated.

The work presented here has following contributions. 1) A deep learning model based on hybrid architecture is proposed. 2) Use of source separation technique to get vocals from audio. 3) Removing the silent parts which accumulates as a noise. 4) Using raw spectrograms to predict the raga from audio. 5) Overlapping windows in generation of spectrograms and sequences.

The rest of the document is structured as follows. The next section of Literature Survey briefs the previous work done in the similar problem domain. In the section Proposed Methodology, the novel approach is presented with Data Preprocessing and hybrid model. The Experimentation section details the experiments conducted during this research work. The results and findings of experiments are presented in the Results and Discussion section. The final section of Conclusion concludes the output of the work. The bibliography is listed at the end.

2 Literature Survey

In a conventional way, the task of *raga* recognition has been addressed during both learning-teaching phase and musical performances. Pupils of Indian Classical Music are taught various ragas by scholars in traditional *guru-shishya* way. Listeners of Indian Classical Music recognise the ragas from a musical phrase or a set of musical phrases called *pakad*. This *pakad* contains melodic theme of a raga encapsualtes the essence of that raga. One can identify raga by listening to the *pakad*. However, for beginners, it is very difficult to identify the raga from *pakad* and requires very high attention of the listener.

Being a very key element of Indian Classical Music, multiple attempts have been made to recognise the raga from the audio. As mentioned earlier, the ragas are ordered sets of *svaras*. Also, svaras are computationally very basic feature to extract from the audio. In several works including [1], authors extracted the svaras from the audio and then matched them with the already known svaras of the ragas. This method works on very basic features i.e. the svaras. Though it can capture the melodic details of music till some extent, it does not consider the temporal details of the same. Further, some ragas in the Indian Classical Music have same set of svaras but they are different ragas. In the works [2–5], authors used ascending and descending progressions of svaras called *arohana* and *avrohana*. Using arohana and avrohana, authors captured the temporal aspects of the music. They generated discrete representation of the melody using quantized pitch vectors. Even though it is a very simple and fast approach in raga recognition, it is not very accurate because of the discrete representation.

The work of Sarkar et al. [14] also used *vadi*, *samvadi*, *arohana* and *avrohana* of raga to classify them. The svara profile based on pitch is formed. The audio was then split into frequency bands and a magnitude spectrogram was generated. The audio was then represented as 168 dimensional vector and used with SVM classifier. Another novel approach was by Padmasundari et al. [15] in their work to identify the raga using Locality Sensitivity Hashing. By extracting pitch vector and doing locality search in its neighbourhood from the known database. The approach is highly scalable and can identify up to 3000 ragas. However, the accuracy is affected and the similar ragas cannot be identified very well.

In the work of Gulati et al. [6], authors used vector space model to represent the audio signals. Similar to vector space models in the field of Natural Language Processing, they built a predictive model to classify the ragas. They obtained 70% accuracy in classifying 40 ragas and 92% accuracy in classifying a subset of 10 ragas. Another work of Gulati et al. [7] showed even better results by using their Time-Delayed Melody Surface (TDMS). Authors performed pre-processing steps including Predominant Melody Estimation and Tonic Normalization followed by surface generation to generate TDMS. The works [6,7] were the first to make their CompMusic dataset publicly available and open the opportunities of research and application of data-driven approaches in Music Information Retrieval for Indian Classical Music.

The applications of artificial neural networks is vast and Music Information Retrieval is no exception. Convolutional Neural Networks (CNN) are proven to be effective in capturing patterns in images and time series data. In [8], author used CNN to recognise the ragas on the CompMusic dataset [6,7]. Using the grayscale plots of pitch vectors as inputs to a CNN model, author achieved accuracy of 96.7% on the set of 5 distinct ragas and 85.6% on the set of 11 distinct ragas. However, the accuracy of the model was around 50% when tested on allied ragas, the ragas which are similar to each other.

In another work titled PhonoNet [9], author used a multi-stage deep neural network model. In PhonoNet, audio clips are converted into mel-Chromagrams. Short Time Fourier Transform was used because of *taans* in Indian Classical Music. The first stage, CNN was trained to detect the ragas on CompMusic dataset fed in smaller chunks. In the second stage, the fully connected layers at the end of CNN was replaced by LSTM cells Layer. The motivation behind this CNN-LSTM hybrid model is that the CNN when trained first will learn the specific filters to identify the nuances of the raga. The second stage LSTM trained on larger chunks will learn the temporal melodic details of the audio and will be able to identify the raga. After training both stages along with data augmentation, PhonoNet achieved state of the art accuracy.

Similar to the Word Embeddings in Natural Language Processing, J. Ross et al. [10] proposed identification of similar ragas from embeddings which they termed as *note-embeddings*. These embeddings were learnt from the notations. Authors used the *bandish* notation of the compositions to train the recurrent network and learn the note-embeddings. Even though these embeddings cannot directly identify a raga from an audio recording, the embeddings can be used to detect the similarity between two recordings in terms of raga. The cosine similarity between two vectors can be a useful parameter in MIR.

The latest SOTA approach to the raga recognition problem on the CompMusic dataset is given by [11]. This was the only paper where the audio was preprocessed using audio source separation techniques. The authors separated vocals from the mixed audio using Mad Twinnet [11]. After performing tonic normalization and pitch tracking, the pitch values were fed to an LSTM recurrent neural network with Attention layers. The model named "SRGM1" achieved state of the art results on the CompMusic dataset with the accuracy of 97% on 10 Ragas subset. The authors also created another model "SRGM2" which solved the problem of sequence ranking.

3 Proposed Methodology

The recognition of raga from an audio is complex problem and can be tackled by signal processing and deep learning methods. Using the appropriate preprocessing, the audio files can be molded into sequences of some form interpretable by deep learning models. Thus, the problem of raga recognition can be transformed into a problem of a sequence classification. The proposed method takes an input audio and preprocesses it using the Data Preprocessing pipeline explained next.

It then feeds the output of the preprocessing to the deep learning model. The output of the model is a prediction of the raga for the input audio.

3.1 Data Preprocessing

Data Preprocessing can play a huge part in the performance of the whole system. The main aim of the preprocessing steps is how to best represent the audio data so that the deep learning models can extract features easily. A data preprocessing pipeline has been proposed which takes in the raw audio as its input and outputs spectrogram images. Figure 1 shows the detailed block diagram of the proposed data preprocessing pipeline. The various components in the preprocessing pipeline are explained below.

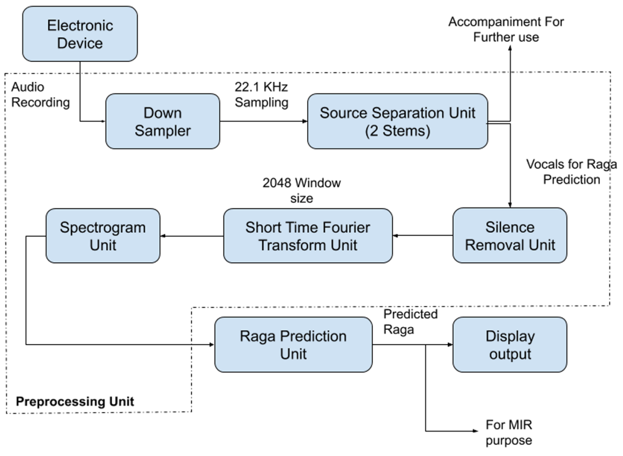

Fig. 1. Data preprocessing pipeline

Down Sampler. Audio files are recorded at some standard frequency, usually 44.1 KHz or 22.1 KHz. In order to have all the audio files at same frequency, the down sampler samples the input audio to 22.1 KHz. This down sampler is not required if the audio files are already at 22.1 KHz.

Source Separation. Audio source separation is a very active area of research. Source Separation unit separates out vocal track of a performance from the entire audio track. By performing source separation, the output of the data preprocessing unit will be more filtered. Another point to consider while performing source separation is that in an Indian Classical Music performance, the vocals of the singer are sufficient to determine the raga of the audio. While there is a loss of useful data of accompanying instruments like harmonium and *sarangi*, the data is more clear and the model should be able to extract features much easily.

Figures 2 and 3 demonstrate the difference by performing source separation. Figure 3 is more clear with the disappearance of the *tanpura* in the audio.

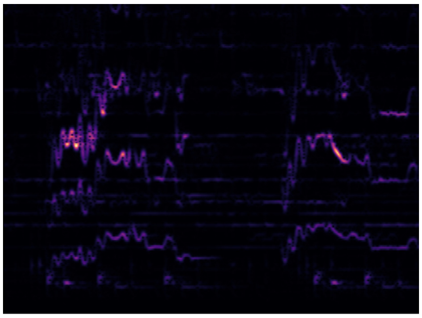
Fig. 2. Spectrogram image without source separation

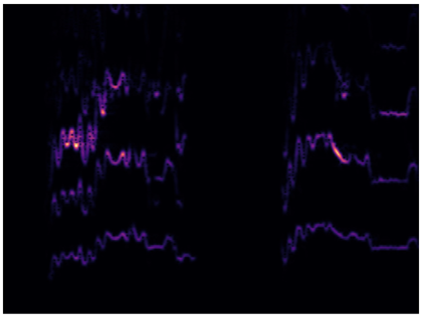
Fig. 3. Spectrogram image with source separation

Silence Removal. Although silence plays an important role in Indian Classical Music, an empty spectrogram is not much of use to predict the raga. This unit removes the silent parts of the audio. There are two methods used in this unit out of which at a time only one is used. The first method is to use InaSpeech Segmenter [12] which uses deep learning to predict silent parts of audio along with other things. The second method is to use simple thresholding. Any part of the audio where the intensity drops below 60 dB is automatically considered as silence.

Short Time Fourier Transform. The audio received from the previous block is fed to the Short Time Fourier Transform Unit (STFT Unit) which performs the Fourier Transform on the audio signal. This translates the audio from time domain to the frequency domain. It is here that one has to strike a balance between resolution in time and frequency. The window size and the overlapping percentage are some of the hyperparameters experimented on in this work. For example, an experiment can be done with *n fft* 2048 and 50% overlapping.

Spectrogram Unit. The output of the STFT unit is the input to the Spectrogram unit, which generates a spectrogram image of the audio. This unit plays an important role in deciding the representation of audio data as images. Changing the color scales, the y-axis scales, the y-axis limit can all change how deep learning models can easily extract features from the images.

All these units together form the Preprocessing unit. All units can be dropped except the STFT unit and the Spectrogram unit depending on the experiment.

Further experiments are carried out using this preprocessing pipeline in different modes.

3.2 Model Architecture

The data processed using the preprocessing pipeline is fed to the model to get the prediction about the raga. The earlier discussion showed the need of the deep learning model which can extract the features from the spectrogram and then identify the sequences of these features. The Convolutional Neural Networks (CNN) are exceptionally good at extracting the features and can represent them as a single vector. The Long Short Term Memory (LSTM) Cells can learn and identify the sequences. Thus, an hybrid of CNN and LSTM is very suitable for this problem statement. The architecture of the proposed model which consists of CNN and LSTM models is detailed further.

Convolutional Neural Network. The last step of the Preprocessing pipeline is the Spectrogram Unit which converts the audio into raw spectrogram images using the output of Short Time Fourier Transform Unit. These images are 324 pixels in width, 216 pixels in height and RGB in colors. The amount of audio data which is represented in one such image can be tuned easily, say 5 s or 10 s. The Fig. 4 shows the architecture of the CNN model with the dimensions for each layer labeled.

The input image in the form of a tensor of shape <3, 216, 324>(Channels, Height, Width) is fed to the CNN model. Convolution layer with kernel size 3, Batch Normalization followed by ReLu activation, Dropout and Max-Pooling layers are applied. The Max Pooling layer in the initial layer has kernel size and stride of 3 each while for next layers it is 2. The same sequence is applied 3 more times to get the final tensor of shape <64, 9, 13>. This tensor is then flattened into a single vector of shape 7488. This vector represents the features of an input and hence called as a feature vector. A fully connected hidden layer of 512 neurons is used after this. Then, a ReLU activation and dropout is used. This fully connected layer is then finally connected to the output layer of size equal to the number of ragas present in the dataset. The Log-Softmax activation is used at the end.

The same architecture is used in all the experiments in the Experimentation section. The model is trained separately from LSTM on dataset and then tested for accuracy. Further details on training procedure and hyperparameters is attached with each experiment.

Long Short Term Memory. The CNN model described above gives an output as a prediction of the raga. This prediction is based on only one spectrogram and thus has no information about notes played before that spectrogram. To capture this temporal information, a sequence of spectrograms or their feature vectors is required. In order to get the feature vector from the spectrogram, the last fully connected layers are dropped from the trained CNN model. Thus, a

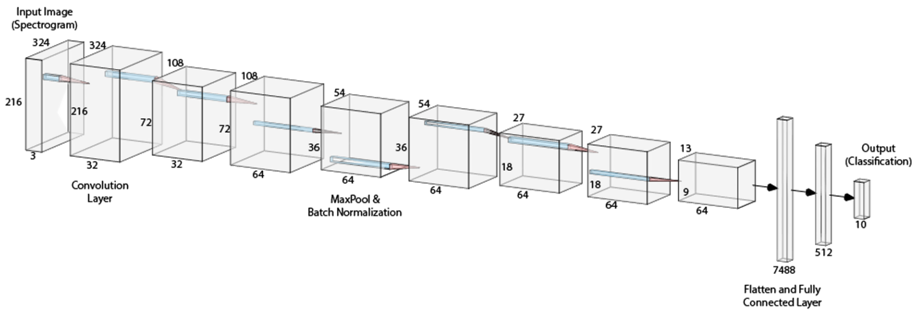

Fig. 4. The architecture of the proposed CNN model

flattened vector of size 7488 is obtained for each spectrogram input fed to the CNN. Using the consecutive spectrograms from the audio and converting them into feature vectors using CNN model, a sequence of feature vectors is obtained.

The LSTM model is trained on these sequences with the weights of the CNN model frozen and backpropagation done only on the LSTM model. The LSTM model with one layer and hidden size of 256 is created. The sequences of three or five feature vectors are used to train this LSTM model with suitable hyperparameters for each experiment.

The data preprocessing and the proposed models are used for all the experiments discussed in the next section. Some of the units in preprocessing pipelines are tuned or dropped or replaced in order to investigate the effects on model predictions. The model architecture stays the same throughout the implementation with slight changes in the hyperparameters.

4 Experimentation

In order to achieve the best possible results, best preprocessing pipeline, optimised training strategy for both CNN and LSTM models and validate our hypotheses, certain experiments were conducted. Each of these experiments used preprocessing and models from the previous section and is detailed further. The results and findings of all the experiments is presented in the following section of Results and Discussion.

The dataset used in this paper is the Hindustani Music Raga Recognition Dataset [7], which consists of around 116.2 h of audio data of 30 commonly heard ragas. There are 10 recordings for each raga in this dataset, though the length of those recording may vary.

It is important to understand the problem definition and train-test splits. For CNN, the problem is framed as identifying raga of n-seconds spectrogram. For LSTM, it is to identify raga of a sequence of spectrograms. The train-validation-test split is done on spectrograms (in case of CNN) and on sequences of spectrograms (in case of LSTM). Another small set of audios which is out of the CompMusic Dataset is used to validate our hypotheses.

4.1 Experiment 1

The goal of this very first experiment was to try out the proposed system on a very small subset of the data. Thus, two ragas, Abhogi and Basant were chosen from the dataset for this experiment as they are structurally different and have roughly the same amount of data. In the preprocessing unit, the vocal part of the audio was separated using Spleeter [13] in the Source Separation Unit and silent parts were removed using InaSpeech Segmenter [12]. The spectrograms were generated by processing the audio file in a stream using the librosa library. The parameters for taking input are frame length of 2048, hop length of 1024 and block size of 256. While performing STFT on the data, the number of output bins *nfft* was 4096, and the hop size was 64. A total of 1194 images were generated in this way. These images were then further split into train, test and validation datasets with a ratio of 80:10:10. A CNN model was trained in this experiment with the batch size of 8 for 500 epochs with the learning rate of 0.00001. The CNN model was able to achieve 96.64% accuracy on the test dataset in classifying these two ragas just by looking at spectrograms.

4.2 Experiment 2

The good results in experiment 1 were indication that CNNs are effective. The question was how scalable they will be with respect to more classes. To test that, we kept all the other parameters exactly the same as the experiment 1 except the number of ragas. The subset of CompMusic dataset was taken with the five ragas, Abhogi, Basant, Alhaiya Bilawal, Ahir Bhairav, Bairagi. A total of 3233 images were generated from these 5 ragas. In order to avoid overfitting, the model was trained for 200 epochs. After training for 200 epochs the model achieved 90% testing accuracy on the test dataset which had 324 images.

4.3 Experiment 3

In this experiment we study the effect of changing hyperparameters by tweaking the *n fft* parameter in the STFT unit to 2048. The intuition behind changing the *nfft size* parameters is that different representation of the same data might affect the way a model learns the distribution of the data. A similar model was trained with this data and we achieved the similar results in 200 epochs. The testing accuracy was 96.6%.

4.4 Experiment 4

As Experiment 3 is similar to Experiment 1 with the *nfft size* cut down to half, this experiment is similar to the Experiment 2 with *nfft size* 2048. There was not any significant improvement in the testing accuracy which was 90.625% compared to 90% of experiment 2.

4.5 Experiment 5

After noticing that the accuracy of CNN dangling around 90% for the subset of 5 ragas, the generation of spectrograms was tuned heavily. With no adjustment in the STFT, the spectograms were generated using the window length of 10 s and the hop length of 5 s. It means, there was 10 s long data in the spectrogram and the consecutive spectrograms had the overlap of 50% data. Due to the overlap, there was a significant increase in the number of images generated for the same amount of ragas, i.e. 7540 images. After training for just 100 epochs, batch size of 8 and the learning rate of 0.001, the CNN model was able to achieve the testing accuracy of 96.81%. This clearly outperforms the models in experiment 2 and 4.

4.6 Experiment 6

To expand more on the complexity of the dataset, five more ragas (Bhup, Kedar, Bilaskhani Todi, Puriya Dhanashree, Rageshri) were added to the subset. The choice of ragas is very interesting here. Raga Basant and Raga Puriya Dhanashree have the same notes and are melodically very close to each other. All the selected ragas had roughly the same amount of data to avoid class imbalance. The audio files are vocal separated and silent parts are removed using InaSpeech Segmenter [12]. Spectrograms are generated with a window size of 10 s and a hop size of 5 s. The parameters for the STFT algorithm are *nfft* 4096 with hop of 64 frames. The total number of images generated in this experiment is 15854 which were split into 80:10:10 ratio for training, validation and testing. Since this experiment was a scaled out version of experiment 5, we increased the number of epochs back to 200 with rest of the hyperparameters same. The trained model achieved the testing accuracy of 96.01%.

4.7 Experiment 7

This experiment was same to previous one, the only difference being in the silent separation unit. The previously mentioned method classified some data as noise and silent parts due to the varying quality of background noises in the audio. Simple threshold based silence removal is used in this experiment. The CNN model was set to train for 100 epochs but the best validation score was achieved before 100 epochs. The best model saved had achieved a validation accuracy of 94.8%. On the test dataset, the best model achieved the testing accuracy of 94.5%. In this experiment we used the best performing CNN model and trained the LSTM model to classify sequences of feature vectors. Sequences of length 5 were generated from all images. These sequences have to be consecutive in nature, i.e. the audio represented by the images must be in one consecutive sequence. Also, the sequences cannot span over multiple audio files, as that would not be a valid input. A total of 18,661 sequences were created in such a manner. The batch size for training sequences was 16 and a learning rate of 0.0001. LSTM model was trained for 20 epochs with a best validation accuracy of 91.4%. The best performing model achieved a testing accuracy of 98.06%.

4.8 Experiment 8

The aim of this experiment was to analyse the effect of vocal separation on the accuracy of the deep learning models. Therefore the vocal separation unit in the data preprocessing pipeline was skipped, and the models were trained on images directly generated from the raw audio with other hyperparameters same. Batch size was selected to be 8 and the learning rate was 0.0001. The model after 100 epochs achieved a testing accuracy of 99.78%. The Fig. 5 shows the loss vs epoch curve for training of the CNN in this experiment. Similar to the previous

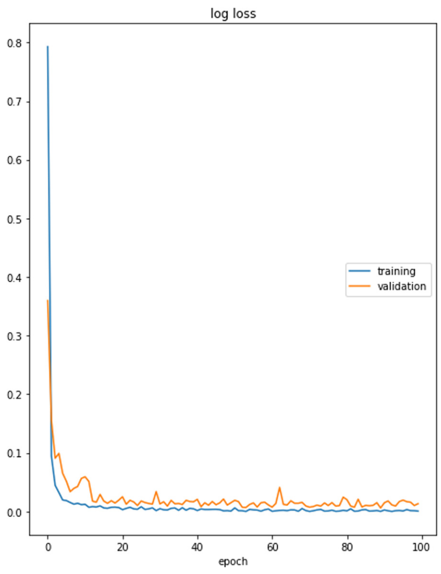

Fig. 5. CNN training in Experiment 8

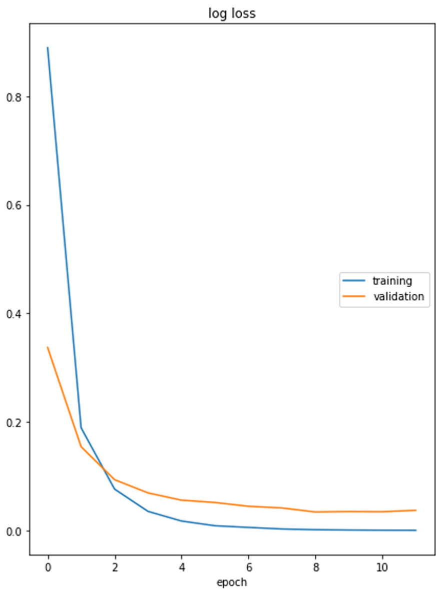

Fig. 6. LSTM training in Experiment 8

experiment, a total of 22,646 sequences were generated. The batch size was 16 and the learning rate was 0.0001. The model was set to train for 20 epochs, but the best performing model was at epoch number 10. Using the early stopping method to prevent overfitting, after the validation loss did not decrease for 3 consecutive epochs, the model was stopped for training. The best performing model had a validation accuracy of 98.7%. It also achieved a testing accuracy of 98.98% on the test dataset. The loss vs epoch curve for LSTM training in this experiment is shown in the Fig. 6.

4.9 Experiment 9

This experiment was conducted to test the scalability of the proposed solution, whether it would scale up to 30 ragas or not. A CNN model was trained on data of all the 30 ragas. Some of the ragas in those 30 ragas were closely related to each other. Some examples being Raga Basant and Raga Puriya Dhanashree, Raga Bihag and Raga Maru Bihag. There were 82868 images generated from around 116.2 h of audio data. The batch size was increased to 16 from 8 and the learning rate was 0.0001. The CNN model was trained for 20 epochs in which it achieved a best validation accuracy of 99.4%. The model had achieved the testing accuracy of 99.51%. Similar to the previous experiments, sequences were generated of sequence length 5 from the available data. A total of 81368 sequences were generated. The LSTM model was trained with a batch size of 16 and a learning rate of 0.0001. The LSTM model after 5 epochs achieved a testing accuracy of 98.4% by correctly classifying 8007 sequences out of 8137 sequences.

Table 1. Table of comparisons of preprocessing steps of the experiments performed.

Expt. no.	Source separation	Silence removal	Spectrogram parameters (*nfft size*, hop size)	Total images	Testing accuracy
1	Yes	Using Ina model	4096, 64	1194	96.64
2	Yes	Using Ina model	4096, 64	3233	90.0
3	Yes	Using Ina model	2048, 64	1192	96.6
4	Yes	Using Ina model	2048, 64	3213	90.625
5	Yes	Using Ina model	4096, 64	7540	96.81
6	Yes	Using Ina model	4096, 64	15854	96.01
7	Yes	Simple threshold	4096, 64	19158	98.06
8	No	Simple threshold	4096, 64	23143	98.98
9	No	Simple threshold	4096, 64	82868	99.51

The table 1 shows the summary of the experiments. The experiments listed here are easily replicable and all the code used by authors is openly available. The plots for each experiment, the experiment notebooks and the inference code is available at this repository https://github.com/dev1911/raga_plus_plus.

5 Results and Discussion

This section briefs the results of the experiments and the findings. The models and preprocessing strategy proposed here work very well on the CompMusic dataset. The experiments conducted show these testing accuracies peaking more than 98%. The results from various hyperparameters can be drawn from these experiments as well. The *nfft size* for STFT Unit was taken 4096 after trying out various combinations. From experiments number 1, 2, 3 and 4 it is very clear that the *nfft size* of 2048 does not improve the model by a very large margin. Even lower sizes will generate distorted spectrograms with more noise. The confusion matrix for the CNN model which was trained with data of 30 ragas is shown in Fig. 7.

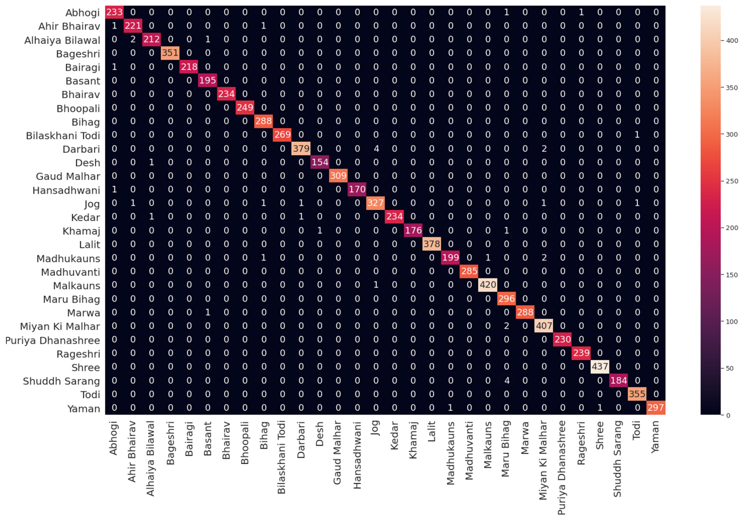

Fig. 7. Confusion matrix

In order to validate the applicability of this approach, some audio files which are completely out of the dataset where taken to test the approach. One of the songs was raga *Bhoopali* sung by Gaansaraswati Kishori Amonkar https://youtu.be/ipauyMfVYso on which both CNN and Hybrid models were tested. The Fig. 8 shows the predictions of the raga by CNN and the Fig. 9 shows the same by Hybrid model. With the better sequence length, the LSTM model will be able to give even better results.

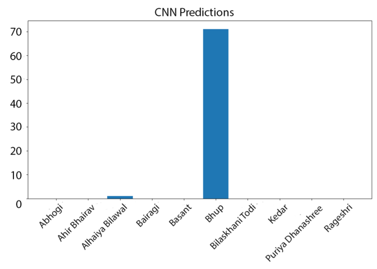

Fig. 8. Predictions by CNN for Raga Bhoopali

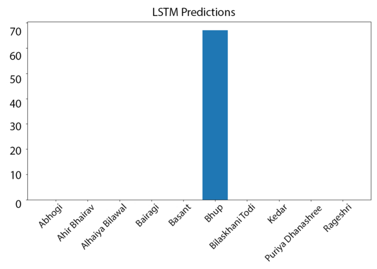

Fig. 9. Predictions by LSTM for Raga Bhoopali

One of our hypotheses is that the models will be able to predict the raga in the slower parts of performances with higher confidence and as the speed of the performance increases, the model might suffer from less confident predictions. To validate that, an audio of raga *Alhaiya Bilawal* by Ustad Rashid Khan https://youtu.be/mGobQ9ztOnY was taken. The models were able to predict the correct raga however the confidence level of the models is very interesting throughout the performance. Figures 10 and 11 shows the confidences of the predictions of the CNN model and LSTM model respectively with respect to time. It can be clearly noticed that when the faster tempo parts of performance start, there is a drastic fall in the confidences of models. As discussed earlier, this is because the features of audio during the fast tempos are not very distinctly clear in spectrograms. Accuracy of these models can be improved by training a different model which is specialised in extracting features from the spectrograms of fast tempo audio.

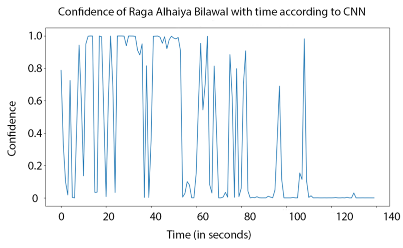

Fig. 10. Confidence of Alhaiya Bilawal vs time - CNN

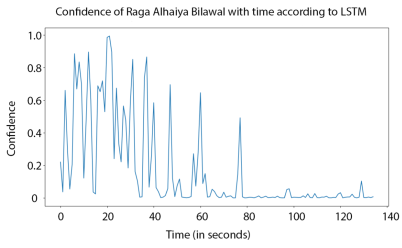

Fig. 11. Confidence of Alhaiya Bilawal vs time - LSTM

Another hypothesis to be validated through these experiments was to study the effect of Sound Source Separation on the performance of the models. Overall, the sound source separation results in better and consistent predictions by model. This was again tested using various songs including the ones which are out of the dataset. As an example, the audio of raga *Puriya Dhanashree* by Pandit Bhimsen Joshi https://youtu.be/AXHMckpzX40 was taken and fed to the same models as it is and also with sound source separation using Spleeter [13]. The bar graph in the Fig. 12 shows the predictions of CNN when the original song was used as the input. The one in the Fig. 13 shows the predictions of CNN when the source separated audio was used. The source separation improves the performance of model by a very large margin.

However, there are cases where the original audio performed better than the source separated counterpart of the audio. Mostly, this is because the models might make predictions by looking at frequency lines of instrumentals like Harmonium or *tabla*. Thus, this hypothesis is not entirely correct at this point. It needs more investigation because the fact that the deep learning models are just black boxes at one end. The separation unit uses Spleeter [13] which is not specifically developed for Indian Classical Instruments yet very decent at it. This

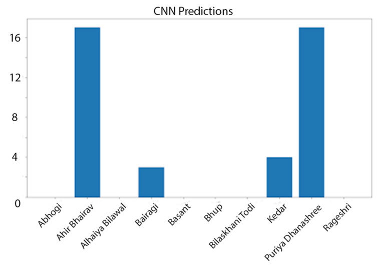

Fig. 12. CNN predictions on original audio of Puriya Dhanashree

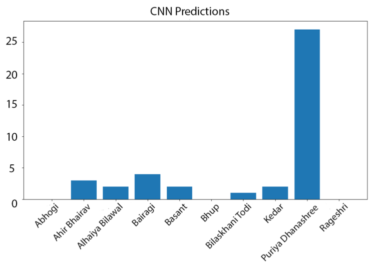

Fig. 13. CNN predictions on source separated audio of Puriya Dhanashree

hypothesis can fully evaluated by using a source separation model trained truly for ICM instruments.

6 Conclusion

Raga being an essential part of Indian Classical Music has a very huge impact on both forms of ICM, Carnatic as well as Hindustani Music. The intricacy of the problem of raga recognition can be tackled using the advancements in Signal Processing and the field of Deep Learning. Using the proposed pipeline of preprocessing and trained models, it is possible to predict the raga from the audio and its raw spectrograms. The use of sound source separation and silent part removal can be very well used along with the Short Time Fourier Transform for the preprocessing. The hybrid architecture of Convolutional Neural Networks and Long Short Term Memory Cells is used to achieve very accurate models. The models can classify the raga from the set of ten ragas with close to 99% accuracy. These models can also identify the allied ragas with better accuracy than most of the previous works.

The proposed hypotheses were validated through the experiments and their results. The accuracy of the model surely decreases during the faster temp parts of the performance. The sound source separation does affect the accuracy of the model. The source separated audio gives better results in most of the cases however the original audio is marginally better in some cases. The use of raw spectrograms works very well in predicting the ragas along with the proposed preprocessing steps.

We would like to express our sincere gratitude to author Devansh Shah's guru Pt. Pradeep Dhond and Mr. Sudhir Parekh.

References

1. Chakraborty, S., De, D.: Object oriented classification and pattern recognition of Indian classical ragas, pp. 505–510 (2012)
2. Pandey, G., Mishra, C., Ipe, P.: TANSEN: a system for automatic raga identification. In: IICAI (2003)

3. Chordia, P., Rae, A.: Raag recognition using pitch-class and pitch-class dyad distributions, pp. 431–436 (2007)
4. Dighe, P., Agrawal, P., Karnick, H., Thota, S., Raj, B.: Scale independent raga identification using chromagram patterns and swara based features. In: 2013 IEEE International Conference on Multimedia and Expo Workshops (ICMEW), pp. 1–4 (2013)
5. Kumar, V., Pandya, H., Jawahar, C.V.: Identifying ragas in Indian music. In: 2014 22nd International Conference on Pattern Recognition, pp. 767–772 (2014)
6. Gulati, S., Serra, J., Ishwar, V., Sentürk, S., Serra, X.: Phrase-based raga recognition using vector space modeling. In: IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Shanghai, China, pp. 66–70 (2016)
7. Gulati, S., Serra, J., Ganguli, K.K., Sentürk, S., Serra, X.: Time-delayed melody surfaces for raga recognition. In: International Society for Music Information Retrieval Conference (ISMIR), New York, USA, pp. 751–757 (2016)
8. Anand, A.: Raga identification using convolutional neural network. In: 2019 Second International Conference on Advanced Computational and Communication Paradigms (ICACCP), pp. 1–6 (2019)
9. Chowdhuri, S.: PhonoNet: multi-stage deep neural networks for raga identification in Hindustani classical music, pp. 197–201 (2019)
10. Ross, J.C., Mishra, A., Ganguli, K.K., Bhattacharyya, P., Rao, P.: Identifying raga similarity through embeddings learned from compositions' notation. In: ISMIR, pp. 515–522 (2017)
11. Madhusudhan, S.T., Chowdhary, G.: DeepSRGM - sequence classification and ranking in indian classical music via deep learning. In: Proceedings of the 20th International Society for Music Information Retrieval Conference, pp. 533–540. ISMIR, Delft, The Netherlands (2019)
12. Doukhan, D., Carrive, J., Vallet, F., Larcher, A., Meignier, S.: An open-source speaker gender detection framework for monitoring gender equality. In: 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) (2018)
13. Hennequin, R., Khlif, A., Voituret, F., Moussallam, M.: "Spleeter: a fast and efficient music source separation tool with pre-trained models. J. Open Source Softw. **5**, 2154 (2020)
14. Sarkar, R., Naskar, S., Saha, S.: Raga identification from Hindustani classical music signal using compositional properties (2017)
15. Padmasundari, G., Murthy, H.A.: Raga identification using locality sensitive hashing. In: 2017 Twenty-Third National Conference on Communications (NCC), pp. 1–6 (2017)