Evolving Systems (2024) 15:1043–1055
https://doi.org/10.1007/s12530-023-09540-x

ORIGINAL PAPER

# Exploiting cepstral coefficients and CNN for efficient musical instrument classification

Saranga Kingkor Mahanta[1] · Nihar Jyoti Basisth[2] · Eisha Halder[2] · Abdullah Faiz Ur Rahman Khilji[2] · Partha Pakray[2]

Received: 23 May 2022 / Accepted: 29 August 2023 / Published online: 14 September 2023


**Abstract**
Identification of musical instruments is a vital problem in the area of Music Information Retrieval. Automatic music classification provides the foundation for a variety of advanced AI applications in the musical domain, such as automatic multi-instrument classification and information extraction from polyphonic audio. In this work, we construct a Convolutional Neural Network (CNN) and train it to classify twenty different musical instrument classes. This work makes use of the Philharmonia dataset, which contains twenty classes of instruments belonging majorly to four instrument families: brass, woodwinds, percussion, and strings. We extract the mel-frequency cepstral coefficients (MFCCs) of the sounds and use them as the input representation for our model. Our work also exploits audio data augmentation techniques on a minority class of the highly imbalanced dataset. To ensure instrumentalist independence, instrument sounds belonging to 14 classes of instruments, that match those from the Philharmonia, are taken from the UIOWA MIS database. The model is trained and tested using this joint dataset. Using the CNN we obtain a new state-of-the-art accuracy.

**Keywords** Audio signal processing · Music information retrieval · Deep learning · Multi-class classification · ConvNets · MFCC

## 1 Introduction

Music is pervasive in our everyday lives. Tunes and music streaming giants profusely categorizing and curating their content material and consequently providing personalized tips and automatic recommendations to the users is an everyday observance in current times (Song et al. 2012). One way of categorization is on the basis of the instruments.

Even competent musicians have difficulties distinguishing between instruments belonging to the same family, as simple as it may appear for humans to differentiate among different musical instruments. This is because many instruments in the same family can have similar timbres or tonal qualities, and the differences between them may be subtle. Additionally, different playing styles, recordings, and other factors can also impact how an instrument sounds, making it even more difficult to accurately identify. The importance of automatic musical instrument classification can be attributed to the need for overcoming such discrepancies and, more importantly, automating an otherwise cumbersome manual task. Thus, it is a difficult but necessary endeavour. Moreover, this forms the basis of more convoluted tasks, namely, music information retrieval, melody extraction, identifying the main instruments in polyphonic audio (Essid et al. 2005), etc. An efficient musical instrument classifier may prove to be very useful in genre detection. Genre determination is greatly dependent on the musical instruments being used to produce a certain piece of music (Fu et al. 2010; Aucouturier and Pachet 2003), for instance, saxophones are highly used in jazz, electric guitars narrow down the range of identification to blues, rock, and heavy metal, and so on.

✉ Saranga Kingkor Mahanta
saranga.mahanta@ip-paris.fr

Nihar Jyoti Basisth
nihar_ug@cse.nits.ac.in

Eisha Halder
eisha20_ug@cse.nits.ac.in

Abdullah Faiz Ur Rahman Khilji
abdullahkhiljinits@gmail.com

Partha Pakray
partha@cse.nits.ac.in

[1] Institut Polytechnique de Paris, Palaiseau, France

[2] National Institute of Technology Silchar, Silchar, India

Furthermore, instrument classification has various potential applications within and beyond the music industry. Within the music industry, it can be utilized for tasks such as automated instrument recognition in audio recordings, facilitating music transcription, improving instrument separation in audio mixing and production, and enabling interactive music systems that respond to specific instruments. Outside of the music industry, this technology can be applied in areas like sound design for films and video games, audio surveillance for identifying specific instruments or orchestration in environmental sounds, etc.

This work proposes a Convolutional Neural Network that expeditiously performs classification on twenty different categories of musical instruments, even across similar families of the same. Furthermore, the distinctive efficiency of MFCCs particular to this task is investigated. In current times, because they address how humans perceive sound, MFCCs are commonly employed for a variety of cognitive tasks involving audio data. This is accomplished by converting the conventional frequency to the Mel scale, which is perceptually relevant to the human hearing system's operation.

The results obtained using the proposed ConvNet on a highly imbalanced dataset, explained in Sect. 3, using some data augmentation techniques pertaining to the lacking class endorses the significance of MFCCs. Previous studies have worked with different subsets of the full Philharmonia dataset, whereas, to the best of our knowledge, our model has trained and achieved the highest accuracy on 20 classes of the entire joint dataset-Philharmonia and UIOWA MIS datasets.

The organization of the remaining paper is as follows: Sect. 2 contains the related works Sect. 3 describes the datasets; the augmentation and pre-processing steps have been explained in Sect. 4; Sect. 5 comprises some relevant theoretical details of the system; the experimental details are elucidated in Sect. 6; the obtained results are elaborated in Sect. 7; Sect. 8 is an extension of 2 and contains performance comparison with other similar works. Lastly, Sect. 9 concludes the study with several potential directions for future work.

## 2 Related works

The task of musical instrument classification has been extensively studied in the past, with various approaches and techniques proposed to tackle the challenge. Initial studies focused on characterizing the timbre of audio signals to measure similarities, including feature selection algorithms such as LDA (Fourer et al. 2014) and automated techniques (Herrera-Boyer et al. 2003). Other studies focused on evaluating different feature schemes for constructing a robust classification system (Deng et al. 2008). The recognition of polyphonic music was also addressed in previous works, including a multi-instrument recognition scheme that built a taxonomy of musical ensembles (Essid et al. 2005)

More recent studies have leveraged the use of Convolutional Neural Networks (CNNs) and Support Vector Machines (SVMs) for identifying musical instruments from audio tracks. The use of cepstral coefficients has also been explored, including the incorporation of phase information into modified group delay features (Diment et al. 2013). Hidden Markov Models (HMMs) were also proposed for musical instrument classification (Eichner et al. 2006). Gaussian Mixture Models and SVMs were studied as classification algorithms (Marques and Moreno 1999).

Deep learning techniques have become increasingly popular for musical instrument classification in recent years. For example, CNNs were used to perform classification on Mel-spectrogram inputs (Haidar-Ahmad ; Solanki and Pandey 2019). Other studies have used multi-resolution recurrence plots (MRPs) that incorporate phase information from raw input signals (Park and Lee 2015). Fractional Fourier Transform (FrFT)-based Mel Frequency Cepstral Coefficient (MFCC) features were also proposed in Bhalke et al. (2016), achieving an average accuracy of 91.84% on 19 musical instruments.

In our previous work, (Mahanta et al. 2021) a feed-forward neural network was trained across all 20 classes of the Philharmonia dataset only and an accuracy of 97% was achieved on a separated test set of the same. A more extensive survey of similar works along with the datasets used, the methods employed, and the performance achieved can be found in Sect. 8.

## 3 Dataset

A joint dataset was used to train and test the proposed model. It comprised the entire 20 classes of instruments of the Philharmonia dataset (Philharmonia 2023) and the 14 classes of the University of Iowa Musical Instrument Samples (UIowa MIS) database (Iowa 2023), namely- bassoon, cello, clarinet, double bass, flute, french horn, oboe, percussion, saxophone, trombone, trumpet, tuba, viola, and violin. These 14 classes of instruments correspond to those of the Philharmonia dataset. Additionally, it has 6 classes- banjo, bass clarinet, contrabassoon, cor anglais, guitar, and mandolin. Our work aims to demonstrate the efficiency of CNNs and basic audio augmentation techniques that aid musical instrument classification. There were a total of 13,679 samples split non-uniformly among 20 classes of musical instruments when the Philharmonia dataset was downloaded. Together with the UIowa MIS, we got a total of 15,016 sounds.

The samples for each instrument class, with the exception of 'percussion', are musical notes ranging from A1 (first octave's note A) to G7 (seventh octave's note G). The tones were played with various dynamics, such as pianissimo, forte, and fortissimo as well as playing techniques, trills, and sustains, such as staccato, legato, vibrato, tremolo, pizzicato, and ponticello, providing us with a diverse set of samples. The dataset is severely skewed, with 1,592 violin examples and only 74 banjo examples at the extremes. There are also 39 sub-classes of percussion instruments in the percussion class, including Agogo Bells, Banana Shaker, Clash Cymbal, Tom-toms, and Vibraslap, to mention a few. Each of these sub-classes has too few examples, i.e ranging from 1 to a maximum of 18 sounds per sub-class, altogether comprising 146 sound recordings of the Percussion parent class of the Philharmonia dataset.

## 4 Data preparation

The provided dataset consisted of noiseless single instrument tones for each instrument class, hence, no complex pre-processing procedures were required before feature extraction. The pre-processing stages that were performed, including handling the class imbalance, are elucidated in the following sub-sections.

### 4.1 Data augmentation

As the sub-classes of the 'Percussion' class have too few individual training examples, we performed data augmentation on the Percussion instruments using the Python Audiomentations package. The Audiomentations (Jordal 2023) library's classes were used with various values of probability parameters (parameters mentioned below) to generate a broad range of audio recordings that differed in different aspects of the sound signal depending on the probability value. To augment the data, the following classes were employed:

- **TimeStretch**: The signal is extended along the time axis without affecting pitch. TimeStretch(min_rate = 0.5, max_rate = 1.3, p = 0.6, leave_length_unchanged=True)
- **PitchShift**: The sound's pitch is shifted without affecting the pace. PitchShift(min_semitones = − 5, max_semitones = 7, p = 0.9)
- **Shift**: The samples are moved forwards or backward, with or without rollover. Shift(min_fraction = − 0.5, max_fraction = 0.5, p = 0.4)
- **Trim**- The leading and trailing silence of a sound signal are removed. Trim(p = 0.7)
- **Gain**- The audio is multiplied by a random amplitude factor to increase or reduce the volume. Gain(p = 0.5)

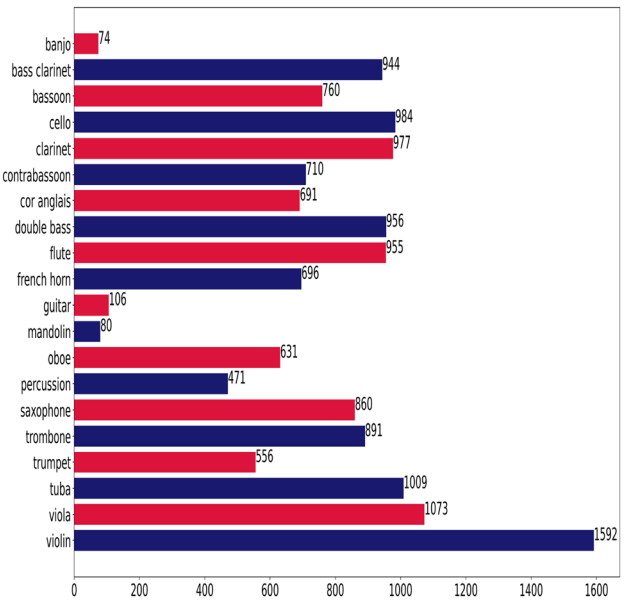


**Fig. 1** Class distribution of the joint dataset

Augmenting audio data can introduce synthetic or modified samples that may not accurately represent the characteristics of the original class. If the augmentation techniques used are not carefully chosen or applied, they can distort the underlying information in the audio signals, making it harder for the model to learn meaningful patterns from the data. Thus, augmentation techniques that have minimal impact on the underlying information of the audio signals were considered. The above methods within reasonable limits were chosen to ensure that the augmented samples still retain the essential characteristics of the original class.

Thus, 312 audio files were generated from the 146 files that were initially available from the Philharmonia dataset, and a total of 438 examples were now present inside the Percussion class. Albeit the classes 'Banjo' and 'Guitar' had comparatively fewer examples, they did not consist of further sub-classes with too few examples each, like in the case of Percussion. Hence, data augmentation was only performed on the latter. The addition of sounds from the UIowa MIS database was done not only as a means of using more of the available data but also to ensure the independence of instrumentalists, i.e multiple instrument players for each instrument class in order to prevent the system from classifying instrumentalists.

Figure 1 shows the class distribution of the partially augmented and joint dataset.

### 4.2 Selecting a constant duration

The durations of the audio inputs across the joint dataset are between 0.078 s to 77.06 s. It is critical to pick a proper length

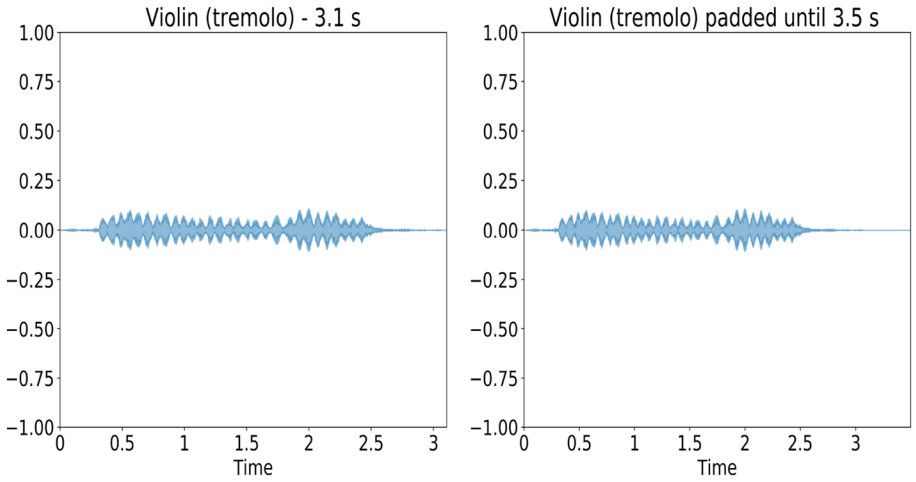


**Fig. 2** An average duration sound clip

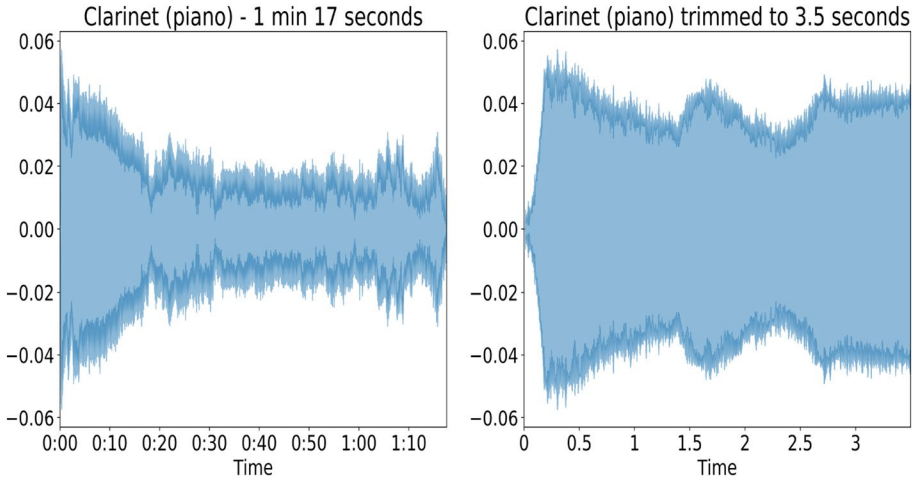


**Fig. 3** Longest audio clip trimmed to 3.5 s

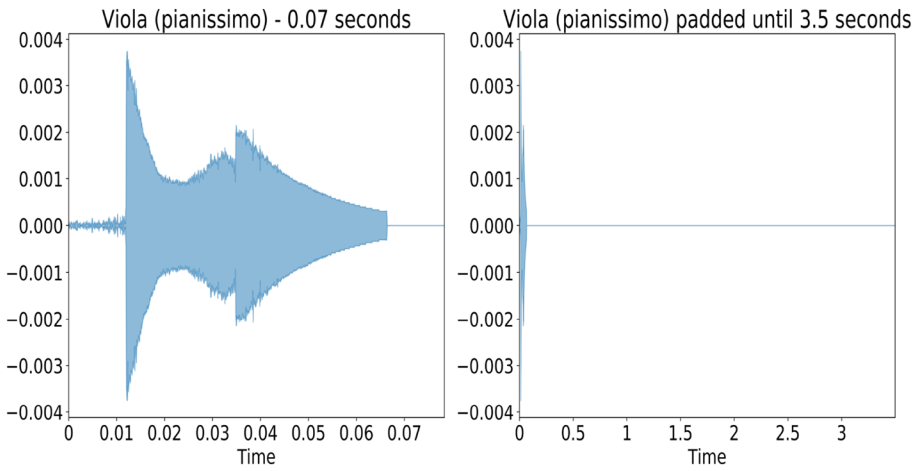


**Fig. 4** Shortest audio clip padded until 3.5 s

for all of the recordings. The sound would be trimmed of essential information if a short duration is chosen. Choosing the maximum duration, i.e. 77.06 s, on the other hand, would result in a large number of sparse values in the inputs. We picked 77,175 samples for each case as the constant number of samples for all the instances, which is comparable to 3.5 s when sampled at 22,050 samples/second. The duration of 13,320 of the total 15,016 recordings is less than 3.5 s. The sound recording in Fig. 2 has an average duration. Figures 3

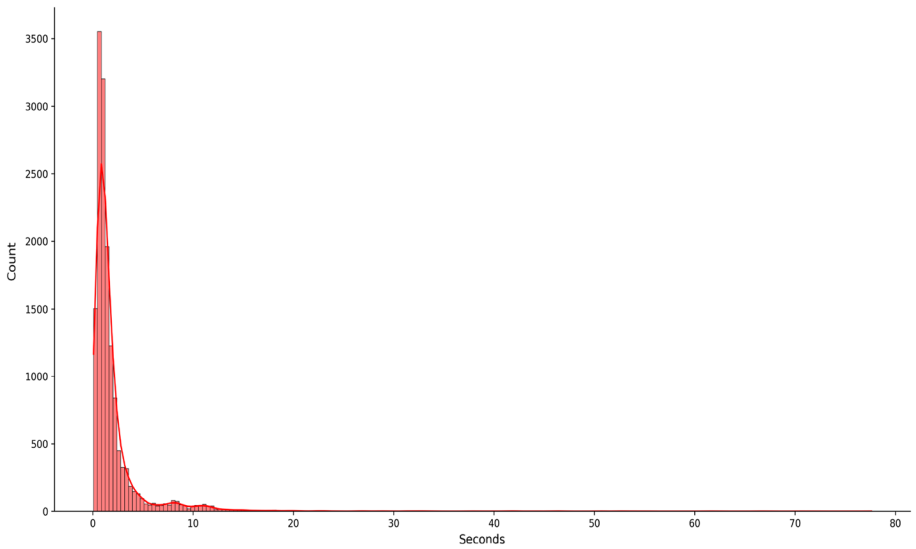


**Fig. 5** Distribution plot of the durations of the joint dataset

and 4 illustrate trimming of the longest sound and padding of the shortest sound of the dataset respectively.

Since all of the audio files' onsets and attacks of sound occur within the first 3.5 s, after which the sound sustains and decays (except for most examples from the Percussion class which consist of only the attack) it may be inferred that no essential information was lost from the trimmed samples. In other words, most of the instances are virtually periodic with small periods, and the identifying aspects of the audio clip take place within the first 3.5 s of the signal. A further rationale for not selecting a constant length of more than 3.5 s is to restrict the number of sparse values generated by padding and to decrease the dimensions. Most importantly, it can be observed from the distribution plot in Fig. 5 that the graph peaks at around 3–4 s. Thus, the first 3.5 s is justified as a suitable duration for all sound recordings.

### 4.3 Feature extraction

Each musical sound file had 15 MFCC coefficients derived from each frame. 2048 samples were contained in a single frame of an audio file. The framing window was created with a hop length of 512 non-overlapping frames. MFCC matrices with size of 151x15 were created using 3.5-second sound samples. These matrices were then unsqueezed to convert the dimension to 151x15x1, which was then fed into the ConvNet, since they expect three-dimensional inputs. The MFCC matrices were the only inputs to the model. A brief explanation of MFCCs is provided in Sect. 5.1.

## 5 System description

The proposed method consists of two fundamental steps: extracting MFCC features from sound sources and putting them into our CNN model for prediction.

### 5.1 MFCCs

MFCCs, spectrograms, and wavelets are commonly used feature extraction methods in audio signal processing, each with its unique characteristics. Spectrograms provide a time-frequency representation of the audio signal, capturing both spectral and temporal information. Wavelets offer a multi-resolution analysis, decomposing the signal into different scales and time-frequency resolutions. In contrast, MFCCs focus on capturing the perceptually relevant spectral features, providing a compressed representation with reduced frequency resolution. MFCCs are often used in speech and music analysis tasks, prioritizing perceptually important features, while spectrograms and wavelets may be more suitable for applications requiring fine-grained time-frequency analysis and preservation of temporal information.

We require our model to effectively differentiate the timbral components of tones belonging to different instruments. Timbre, or the colour of sound, is a characteristic that allows sounds of similar intensity, frequency, and duration to be

discriminated. It also helps to differentiate between different instruments in the same instrument family. Since many audio inputs play similar notes with comparable intensity, the timbre becomes a critical aspect to discern among them. Timbre is primarily determined by dynamic characteristics like harmonic content, the sound envelope like the attack-decay-sustain-release envelope of the sound, and also the modulation. According to (Terasawa et al. 2005) the MFCC representation forms a relatively better model of timbre space. They state in the work, "The 13 colors in the MFCC representation allows for more accurate timbre interpolation and creates a model where the parameter axis are orthogonal than the other representations." MFCCs are, thus, capable of identifying the formants and timbre of sound whilst rejecting background noise (Chakraborty and Parekh 2018). The reason for having 13 MFCCs is based on practical considerations. Having 13 coefficients provides a good balance between the accuracy of the representation and the computational efficiency of the calculation. A higher number of MFCCs would result in a more accurate representation of the audio's timbre, but would also increase the computation time and complexity. On the other hand, a lower number of MFCCs would result in a less accurate representation but would be faster to compute.

The cepstrum of an audio sign indicates the rate of change within its spectral bands. It expresses the unique values that make up a sound's formants and timbral aspects. Equation 1 may be used to get the cepstral coefficients.

$$C(x(t)) = F^{-1}(log(F[x(t)])) \tag{1}$$

Peaks are found by calculating the log of the Fourier transform magnitude of the audio after computing its spectrum using a cosine transformation. The ensuing spectrum lies within the quefrency domain (Oppenheim and Schafer 2004). Amplitude is perceived in a logarithmic fashion by humans. The idea that the Mel scale, a logarithmic scale, is based on is that for equal distances on the dimensions, the perceptual distance is the same. Mel scaling is applied to it using Eq. 2 on frequencies measured in Hertz.

$$Mel(f) = 2595 * log(1 + f/700) \tag{2}$$

Small changes in speech can be detected by humans at lower frequencies. The Mel Scale measures these minute differences in order to draw parallels with human hearing. In many cases, a discrete cosine transformation is used instead of an inverse Fourier transform. Due to the fact that the former gives real-valued coefficients and also decorrelates energy in distinct mel bands, it is preferred. To extract the MFCCs from a frame of audio, the following steps are typically taken:

- Pre-emphasis: A high-pass filter is applied to the audio frame to emphasize higher frequencies and reduce the impact of low frequencies.
- Window function: A window function, such as a Hamming window, is applied to the pre-emphasized audio frame to reduce spectral leakage.
- Fast Fourier Transform (FFT): The windowed audio frame is transformed from the time domain to the frequency domain using the FFT.
- Mel scaling: The magnitude spectrum obtained from the FFT is converted to the Mel scale using Eq. 2.
- Discrete Cosine Transform (DCT): The logarithm of the Mel-scaled magnitude spectrum is transformed using the DCT, resulting in the MFCCs.

### 5.2 ConvNets

A CNN is a class of neural networks whose main components are the convolution layers. The convolution operation is performed by kernels. These kernels or filters detect and extract both low and high-level feature patterns from images. They slide over the input image at a particular stride and perform the convolution operation which is basically the dot-product of the weights of the kernel and a restricted portion of the receptive field (pixel values of the image portion that is superimposed by the kernel at that moment). Edge detection, blur, sharpening, and other actions can be performed by convolving a picture with multiple filters. The kernel weights that are necessary to perform such operations are learned during the training of the model. The dimensions of the input are shrunk after the convolution operations. If the shrinkage of the input data dimensions is not wanted, padding can be performed on the data before the convolution operations to maintain the original height and width.

For an input having height, H and width, W the output dimensions can be determined using Eq. 3:

$$O_h \times O_w = \left\lfloor \frac{H + 2P_h - F_h + 1}{S} \right\rfloor \times \left\lfloor \frac{W + 2P_w - F_w + 1}{S} \right\rfloor \tag{3}$$

where $P_h$ and $P_w$ are the padding amount along the height and width respectively, $F_h$ and $F_w$ are the kernel filter dimensions, and $S$ is the stride size. It is mandatory for the number of channels in the filter to be equal to that of the input. Each filter outputs a 2D feature map having dimensions $O_h \times O_w$. The feature maps resulting from various kernel filters performing different feature operations are then juxtaposed to form an output volume having dimensions $O_h \times O_w \times N$, where $N$ is the number of filters applied, which is actually a hyperparameter of the CNN model. Activation functions are applied to the data points in the output volume to introduce non-linearity, similar to those used in the Dense layers. In our work, we choose the

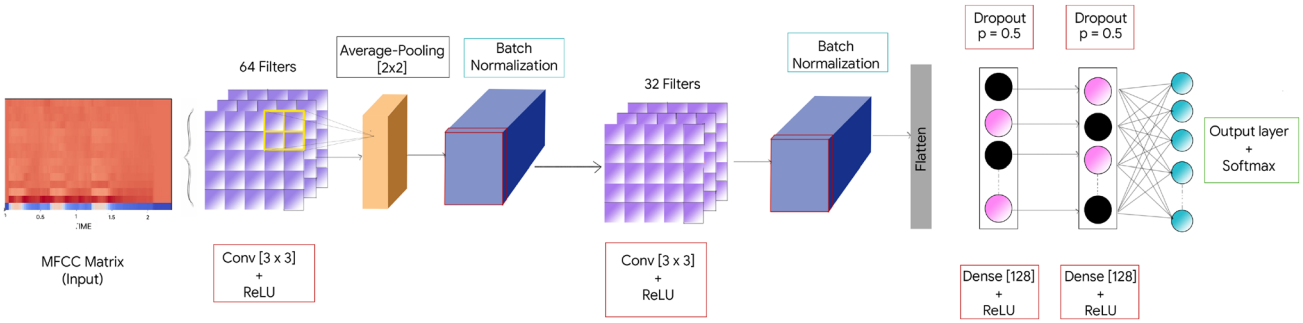


**Fig. 6** Proposed CNN architecture

conventionally used ReLU activation function, Eq. 4, for both the convolution layers and the fully connected dense layers. The model is described in detail in Sect. 6.

$$y = max(0, x) \tag{4}$$

Convolution layers are generally followed by pooling layers. Spatial pooling, also termed as subsampling or downsampling reduces the dimensionality of each feature map of the output volume. This basically helps to reduce the number of parameters of each filter to help speed up computation. Additionally, the most important information in the data portions is retained which increases the robustness. We use Average Pooling layers in our work, which take the average of the elements from the rectified feature maps. There are no learnable parameters in Average Pooling that need to be trained during backpropagation.

Thus, a convolutional block consists of convolution layers usually followed by pooling layers. The input data passes through one or more such blocks sequentially after which the final output volume is flattened. Batch Normalization layers are also often used in these blocks since they normalize the data points with respect to the batch. It helps to attain convergence faster and thus, speeds up training. The flattened data points then flow through an output layer with the same number of neurons as classes, in our case, 20. Fully connected linear layer(s) may be optionally used before the output layer. The Softmax function is applied as the output layer activation function since our goal is a multi-class classification. Using Eq. 5, it calculates the confidence scores of each class. For a given input, the model's predicted class is the one with the greatest confidence score.

$$\sigma(z)_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \tag{5}$$

CNNs learn to detect the necessary and sufficient patterns that define an image without the need to manually extract features for it. Unlike ordinary feed-forward networks which only accept vectors as inputs, CNNs accept 3D data as inputs. The number of learnable parameters is reduced in CNNs since only the kernel weights need to be learned instead of the weights connecting to each data point while training. Furthermore, CNNs help to preserve locality. Since the input data is not flattened, the relative positions of the data points are preserved. This is the major reason behind the excellent performance of CNNs for visual tasks.

A visualization of the CNN architecture used in this work is shown in Fig. 6

## 6 Experimental setup

The joint dataset was split into training, cross-validation, and testing sets using stratified splitting, with the number of samples from each of the 20 categories being divided proportionally. Stratification was necessary to avoid a disproportionate division of examples. The training set contained 10,828, cross-validation 1,915, and the test set contained 2,253 examples.

Considering the trade-off between the accuracy of the audio representation and the computational efficiency, we chose 15 MFCC coefficients for each frame of each example recording (audio data point). The information most pertinent to our goal is found at the lower end of the cepstrum's quefrency axis, such as formants, spectral envelope, and timbre. Increasing the amount of cepstral coefficients would raise the model's complexity correspondingly. Since the dataset is relatively small and imbalanced, this may result in a greater variance problem. Using the Librosa (McFee et al. 2015) library, 15 MFCC coefficients were retrieved per frame of each recording. The frame size was 2048 samples. The framing window was created with a hop length of 512 frames. This means that each frame consisted of 2048 samples, and the next frame would start 512 samples after the end of the current frame. This sliding window approach is used by Librosa to capture the local features of the audio signal in each frame. MFCC matrices with the size of 151 $\times$ 15 were created using 3.5-second sound samples. In other

words, a single frame of 2048 samples was converted into a set of 15 MFCC coefficients. A 3.5-second audio recording sampled at 22,050 samples per second will contain 77,175 samples. With a frame size of 2048 samples and a hop length of 512 samples, there will be a total of 151 frames extracted from the 3.5 s recording. This is why 151 MFCC vectors are extracted from the 3.5-second audio file. After performing an unsqueeze operation and adding another dimension, these matrices were fed into our CNN model.

The MFCC feature matrices are visually comprehensible and recent works incorporating ConvNets have yielded impressive results for visual recognition applications. Thus, we propose a CNN with the hyperparameters tuned on the validation set, as shown in Fig. 6 and elucidated below. The set of hyperparameters that gave the optimal validation accuracy was chosen.

1. Convolution layer- 64 filters, kernel size of 3x3 (denoting height and width dimensions), stride of 1x1, and ReLU activation, taking an input shape of 151x15x1
2. Average pooling layer- 2x2 pool size
3. Batch normalization layer
4. Convolution layer- 32 filters, kernel size of 2x2, stride of 1x1, and ReLU activation
5. Batch normalization layer
6. Flattening layer to convert the outputs from the previous layer into a vector
7. Fully connected layer- 256 neurons with kernel, bias, and activity regularizers, and ReLU activation
8. Dropout layer with dropout rate, p= 0.5
9. Fully connected layer- neurons units with kernel, bias, and activity regularizers, and ReLU activation
10. Dropout layer with dropout rate, p= 0.5
11. Output layer- 20 neurons (for 20 musical instrument classes) and Softmax activation function

The model's structural design choices such as the kernel size, pooling size, number of filters, layers, and neurons per fully-connected layer were arbitrary choices with compliance to some conventional norms, like choosing powers of 2 as the number of filters and neurons in a layer.

During training, (Kingma and Ba 2014) was employed with an initial learning rate of 1e-4 as the optimizer. The mini-batch gradient descent with a batch size of 128 was adopted. The sparse categorical cross-entropy loss function was used since the output labels were not one-hot encoded. The training took place over 200 epochs with callbacks that saved the model on obtaining a better validation accuracy after an epoch. Thus, after the training, the model that gave the highest validation accuracy was loaded and used to measure its performance on the untouched test set. The model training and evaluation in this paper were performed in Google Collaboratory using the provided cloud-based NVIDIA Tesla V100 GPU.

## 7 Results and analysis

A Stratified K-fold cross-validation strategy using 10 folds was also employed to compare the performances of the two models on the joint augmented dataset (excluding the test set). The feed-forward ANN gave a mean accuracy of 96.66% and a standard deviation of $\pm$ 0.55%, whereas the mean accuracy and standard deviation exhibited by the CNN was 98.53% $\pm$ 0.40%. Thus, the CNN was finally chosen and tested on a separate test set.

Albeit an admirable score, the model may have been subject to the accuracy paradox (Valverde-Albacete and Peláez-Moreno 2014). This happens when a model incorrectly identifies minority class examples while correctly classifying a relatively large number of majority class examples, resulting in high accuracy. Thus, the accuracy score on the test set is not the best evaluation metric. On the other hand, the confusion matrix is a reliable metric for classification tasks. It gives a better idea of the classification model by displaying the number of classes correctly and incorrectly predicted. The confusion matrix shown in Fig. 7 proves that the case of the accuracy paradox does not arise in our case, since our model classifies all the classes fairly well. The confusion matrix also shows the wrongly predicted labels of the classes. This helps in gaining insights into the similarity between the two classes. For instance, from Fig. 7, three Violin sound recordings were wrongly classified as Viola, accounting for the similarity of the two musical sounds.

The Receiver Operating Characteristic (ROC) curves and Precision-Recall curves are also credible evaluation metrics, especially for imbalanced classification tasks. The ROC curve measures the distinguishing capability of the model among the classes at different threshold values, while the Precision-Recall curve graphically displays the tradeoff between recall and precision for various threshold settings. In Fig. 8 the poor curves of the 'percussion' class are clearly visible. This is quite expected because, unlike the other classes, the dataset had a tiny number of examples from 39 sub-classes of percussion instruments. However, the data augmentation and adoption of a CNN architecture considerably improved the performance of the model with emphasis on the 'percussion' class as can be observed in Fig. 9.

Table 1 and Table 2 display the precision, recall, and F1 score[1] of each class of instruments displaying the performance of the feed-forward network and CNN model respectively on the test set of the joint dataset. The F1 score

[1] Generated using the classification_report class present in the Scikit-learn library.

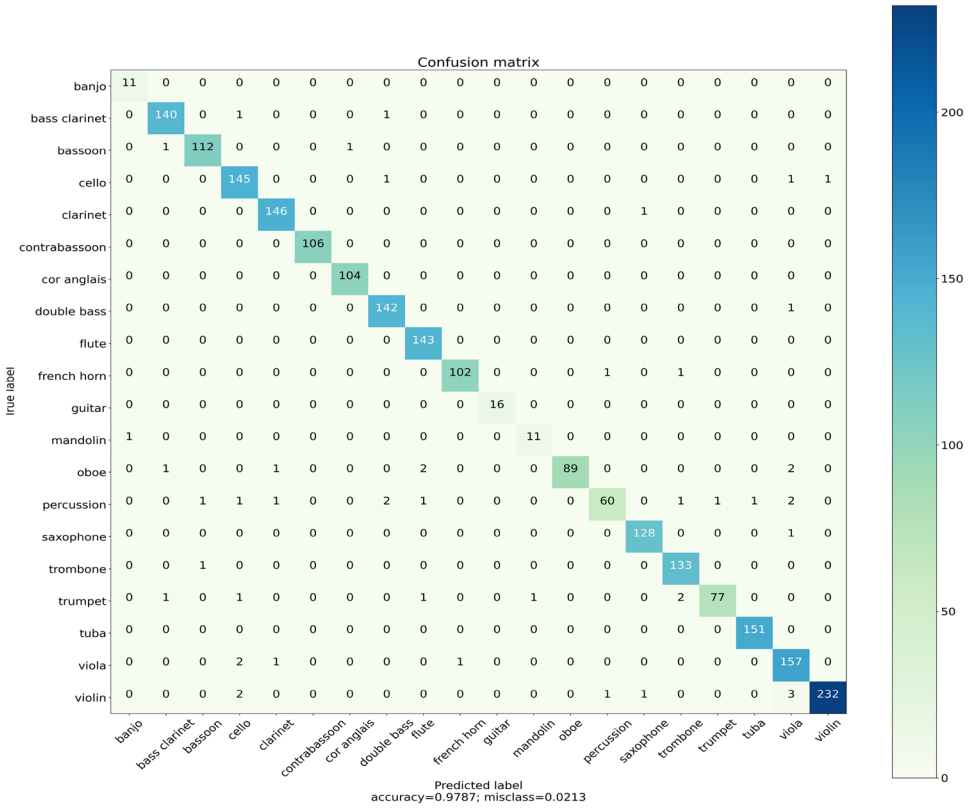


**Fig. 7** Confusion Matrix with respect to the proposed CNN predictions on the test set

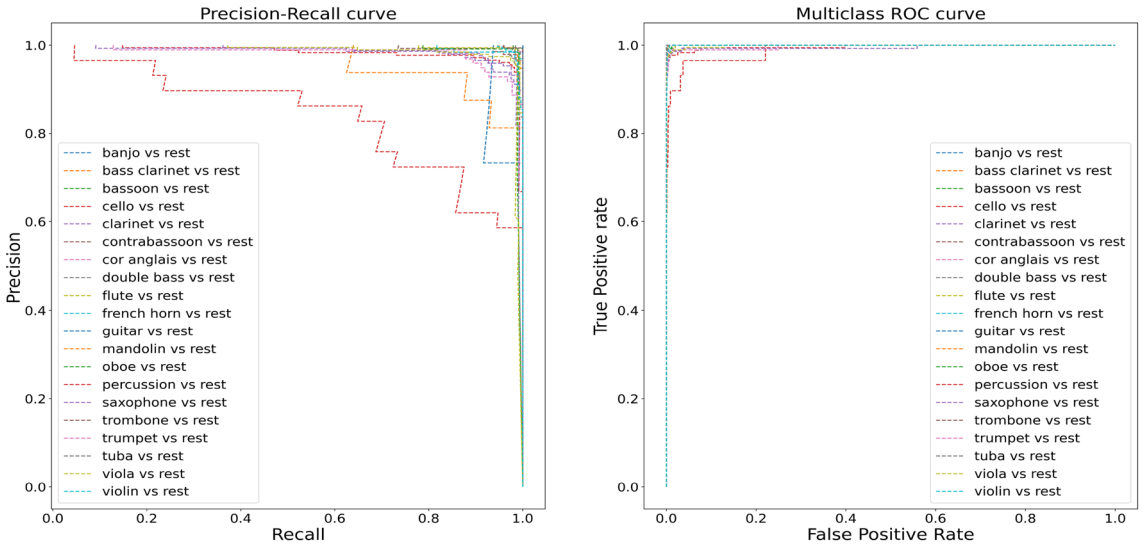


**Fig. 8** Precision-Recall and ROC curves of the feed-forward network

obtained for the 'percussion' class with respect to the feed-forward network model was 0.84. An improvement is seen for the same with respect to the proposed CNN model (0.90). Some of the other classes show a marginal improvement in the F1 scores. The mean F1 score of the CNN was found to be 97.5%, whereas it was 95.15% for the feed-forward network model.

Lastly, an ROC-AUC score[2] of 0.999 was obtained for this particular multi-class classification task with our proposed model.

[2] Generated using the roc_auc_score class of the Scikit-learn library.

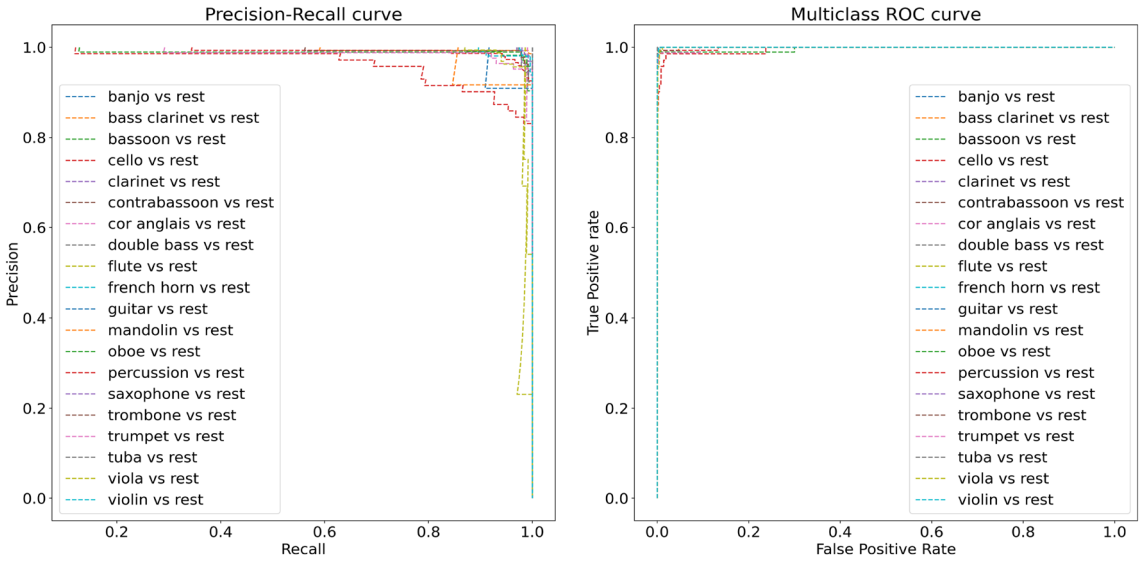


**Fig. 9** Precision-Recall and ROC curves of the proposed CNN model

**Table 1** Classification report of the feed-forward neural network on the test set

| | Precision | Recall | F1-Score | Support |
|---|---|---|---|---|
| Banjo | 0.92 | 1.00 | 0.96 | 11 |
| Bass clarinet | 0.99 | 0.95 | 0.97 | 142 |
| Bassoon | 0.97 | 0.98 | 0.98 | 114 |
| Cello | 0.95 | 0.97 | 0.96 | 148 |
| Clarinet | 0.96 | 0.95 | 0.95 | 147 |
| Contrabassoon | 0.98 | 0.99 | 0.99 | 106 |
| Cor anglais | 0.94 | 0.98 | 0.96 | 104 |
| Double bass | 0.96 | 0.99 | 0.98 | 143 |
| Flute | 0.97 | 0.99 | 0.98 | 143 |
| French horn | 0.98 | 0.97 | 0.98 | 104 |
| Guitar | 0.93 | 0.88 | 0.90 | 16 |
| Mandolin | 0.85 | 0.92 | 0.88 | 12 |
| Oboe | 1.00 | 0.89 | 0.94 | 95 |
| Percussion | 0.90 | 0.79 | 0.84 | 71 |
| Saxophone | 0.88 | 0.98 | 0.93 | 129 |
| Trombone | 0.94 | 0.98 | 0.96 | 134 |
| Trumpet | 0.96 | 0.95 | 0.96 | 83 |
| Tuba | 0.99 | 1.00 | 0.99 | 151 |
| Viola | 0.97 | 0.90 | 0.94 | 161 |
| Violin | 0.97 | 0.98 | 0.98 | 239 |
| Accuracy | | | 0.96 | 2253 |
| Macro avg | 0.95 | 0.95 | 0.95 | 2253 |
| Weighted avg | 0.96 | 0.96 | 0.96 | 2253 |

**Table 2** Classification report of the proposed CNN model on the test set

| | Precision | Recall | F1-Score | Support |
|---|---|---|---|---|
| Banjo | 0.92 | 1 | 0.96 | 11 |
| Bass clarinet | 0.98 | 0.99 | 0.98 | 142 |
| Bassoon | 0.98 | 0.98 | 0.98 | 114 |
| Cello | 0.95 | 0.98 | 0.97 | 148 |
| Clarinet | 0.98 | 0.99 | 0.99 | 147 |
| Contrabassoon | 1 | 1 | 1 | 106 |
| Cor anglais | 0.99 | 1 | 1 | 104 |
| Double bass | 0.97 | 0.99 | 0.98 | 143 |
| Flute | 0.97 | 1 | 0.99 | 143 |
| French horn | 0.99 | 0.98 | 0.99 | 104 |
| Guitar | 1 | 1 | 1 | 16 |
| Mandolin | 0.92 | 0.92 | 0.92 | 12 |
| Oboe | 1 | 0.94 | 0.97 | 95 |
| Percussion | 0.97 | 0.85 | 0.90 | 71 |
| Saxophone | 0.98 | 0.99 | 0.99 | 129 |
| Trombone | 0.97 | 0.99 | 0.98 | 134 |
| Trumpet | 0.99 | 0.93 | 0.96 | 83 |
| Tuba | 0.99 | 1 | 1 | 151 |
| Viola | 0.94 | 0.98 | 0.96 | 161 |
| Violin | 1 | 0.97 | 0.98 | 239 |
| Accuracy | | | 0.98 | 2253 |
| Macro avg | 0.97 | 0.97 | 0.97 | 2253 |
| Weighted avg | 0.98 | 0.98 | 0.98 | 2253 |

## 8 Performance comparison

The task of musical instrument classification has been the subject of numerous studies. One of the earliest attempts was to characterize the timbre of audio signals in order to measure similarities among them and facilitate automatic and semi-automatic database indexing. The process was later automated with the aim of further improving accuracy. Fourer et al. (2014) utilized feature selection algorithms, such as LDA, to classify timbre and achieved a success rate of over 80%. Herrera-Boyer et al. (2003) automated the process of timbre classification to improve accuracy further. Deng et al. (2008) studied various feature schemes that could be used to construct a robust classification system, including human perception-based features, cepstral features, and MPEG-7 audio descriptors. Essid et al. (2005) proposed a multi-instrument recognition scheme that did not require musical source separation and could operate on real-world music. Singh and Kumar used Convolutional Neural Networks (CNNs) (Singh et al. 2019) and Support Vector Machines (SVMs) for instrument classification but failed to distinguish between instruments of the same type. Diment et al. (2013) used a modified group delay feature that incorporated phase information, along with MFCCs, and achieved an accuracy of 71% for 22 instruments. Eichner et al. (2006) proposed the use of Hidden Markov Models (HMMs) for musical instrument classification and created a database of 600 recordings of four instrument types. Marques and Moreno (1999) utilized Gaussian Mixture Models and an SVM as classification algorithms and achieved a 30% error rate in determining the instrument source. Lara Haider utilized a CNN and Mel-spectrograms (Haidar-Ahmad 2019) as inputs to classify audio into four categories (piano, drums, flute, and others) and achieved an F1 score of 64%. Solanki and Pandey (2019) used an eight-layered Convolutional Neural Network and Mel Spectrograms to recognize instruments and achieved an accuracy of 92.8%. Park and Lee (2015) used both Spectrogram images and multi-resolution recurrence plots (MRPs) to classify musical instruments. Bhalke et al. (2016) proposed a classifier model using Fractional Fourier Transform (FrFT)-based Mel Frequency Cepstral Coefficient (MFCC) features and a Counter Propagation Neural Network (CPNN) and achieved an average accuracy of 91.84% on 19 musical instruments.

The complexity of categorization tasks increases as the number of output classes grows. Unlike the prior efforts, which have mostly used only a portion of the total classes in the Philharmonia (except for our previous work (Mahanta et al. 2021)), our model was trained and tested on all 20 classes of the instruments in the combined Philharmonia and UIOWA-MIS datasets. On top of that, we achieved an accuracy of 98% which is, to the best of our knowledge, the state-of-the-art accuracy on this joint dataset. Earlier works have either scored a lower accuracy or a similar or higher accuracy but on fewer instrument classes. A performance comparison of all the works mentioned above along with the datasets used has been summarized in Table 3.

## 9 Conclusion

In this work, we constructed a CNN and trained it on a training set from the joint dataset composed of the Philharmonia and UIOWA MIS datasets. Audio data augmentation techniques were used only in the Percussion class because of the presence of further sub-classes of percussion instruments, which was not the case for the other instrument classes having fewer sound recordings.

Despite the class imbalance and the high auditory resemblance of certain classes with each other, the model achieves an accuracy of 98% in the test set which is, to the best of our knowledge, the state-of-the-art accuracy on 20 instrument classes of the Philharmonia. Previous works have either achieved a higher accuracy on a fewer selection of instrument classes, or a lower accuracy on a greater selection of the same. This can be accounted to the fact that MFCCs being visually comprehensible features of sound, the CNN could perform the classification task considerably well. Accuracy not being the most trustworthy metric for imbalanced classification problems, we also plotted the ROC and Precision-Recall curves. The plots, ROC-AUC score of 0.999, and a mean F1 score of 97.5 confirms the validity of the high accuracy. Our model did not use any pre-trained layers and was completely built and trained from scratch. It is fairly lightweight and thus, can be integrated with real-time web applications with cheaper memory requirements.

Future research on this area has enormous potential. One key area of future expansion might be expanding our target space to include more instrument classes. Our study is limited by the fact that our model is unable to identify between instruments performing in polyphonic audio. It will work effectively for sounds with a single instrument. Considering additional features of sound like mel-spectrograms, spectral centroid, Q-transforms, etc. may also be used as inputs either separately or simultaneously to make better predictions. Instead of using the first certain seconds of the audio examples, the highest average amplitude over a certain time window could be tried. Using standard CNN architectures like VGGNet, Inception, ResNet, etc. could be tried and assessed if they give better performance. The purpose of this paper was to show the effectiveness of a CNN in general.

Furthermore, variations in playing style or technique can have a significant impact on the accuracy of a model's

**Table 3** Performance comparison of mentioned works in Related Works

| Work | Dataset | Performance |
|---|---|---|
| Fourer et al. (2014) | CREM and Iowa databases | >80% accuracy |
| Eronen and Klapuri (2000) | McGill Master Samples collection (30 orchestral instruments) | Instrument family with 94%, individual instrument with 89% accuracies |
| Marques and Moreno (1999) | Custom dataset with 8 instruments | 30% error rate |
| Diment et al. (2013) | RWC Music Database (22 instruments) | 70.7% accuracy |
| Haidar-Ahmad (2019) | AudioSet by Google (3 instruments) | F1 score- 64%, Precision- 70%, Recall- 65% |
| Solanki and Pandey (2019) | IRMAS dataset (11 instruments) | 92.8% accuracy |
| Siebert et al. (2023) | Philharmonia (18 instruments), UIOWA MIS, MIM databases | For Philharmonia, stable precision and recall of 94% |
| Tu and Li (2023) | Philharmonia (8 instruments) | 87% accuracy |
| Deng et al. (2008) | UIOWA MIS | 86.9% accuracy |
| Singh et al. (2019) | IRMAS | Error rate of 6.67% |
| Eichner et al. (2006) | Custom dataset with 4 instruments | 78% correctness |
| Park and Lee (2015) | UIOWA MIS (20 instruments) | Error rates of 1.31%, 6.31% and 6.86% |
| Bhalke et al. (2016) | McGill University Master Sample (MUMS) sound database | 91.84% accuracy |
| Bormane and Dusane (2013) | Custom dataset with 14 instruments | Average accuracy of 72.87% |
| Livshin and Rodet (2004) | Custom dataset of 108 solo performances of 7 instruments | 88% accuracy |
| Prabavathy et al. (2020) | RWC, musicbrainz.org, MINIM-UK, IRMAS, NSynth datasets (16 instruments) | 99% accuracy |
| Lostanlen et al. (2018) | Studio On Line dataset (16 instruments) | 99.7% precision |
| Toghiani-Rizi and Windmark (2017) | Philharmonia (8 instruments) | 93.5% accuracy |
| Chakraborty and Parekh (2018) | Philharmonia and Freesound Music databases (10 instruments) | 93% accuracy |
| Mahanta et al. (2021) | Philharmonia (20 instruments) | 97% accuracy, AUC score - 0.996, average F1 score - 95.9% |
| Our work | Philharmonia (20 instruments) and UIOWA MIS | 98% accuracy Average F1 score- 97.5% AUC Score⁻0.999 |

predictions in instrument classification tasks. Different musicians can exhibit unique playing styles, articulations, and expressive nuances that result in variations in the recorded audio signals. These variations introduce additional challenges for the model, as it needs to learn to differentiate between different playing styles while recognizing the underlying instrument. For example, a guitar played with finger-picking technique will have different spectral characteristics compared to one played with strumming technique. To account for these variations in future research, several strategies can be employed. First, incorporating a larger and more diverse dataset that encompasses a wide range of playing styles and techniques can help the model learn to recognize and adapt to different variations. By exposing the model to a broader spectrum of playing techniques during training, it can develop a more robust representation of the instruments, enabling it to better generalize to different playing styles. It can be beneficial to consider incorporating contextual information in the model architecture. For example, leveraging sequential models such as recurrent neural networks (RNNs) or attention mechanisms can enable the model to capture temporal dependencies and patterns that are characteristic of specific playing styles.

**Data availability** The authors can confirm that all relevant data are included in the article.

## Declarations

**Conflict of interest** The authors declare that they have no conflict of interest.

## References

Aucouturier JJ, Pachet F (2003) Representing musical genre: a state of the art. J New Music Res 32(1):83–93

Bhalke D, Rao CR, Bormane DS (2016) Automatic musical instrument classification using fractional Fourier transform based-MFCC features and counter propagation neural network. J Intell Inf Syst 46(3):425–446

Bormane D, Dusane M (2013) A novel techniques for classification of musical instruments. Inf Knowl Manag 3:1–8
Chakraborty SS, Parekh R (2018) Improved musical instrument classification using cepstral coefficients and neural networks. Methodologies and application issues of contemporary computing framework. Springer, Cham, pp 123–138
Deng JD, Simmermacher C, Cranefield S (2008) A study on feature analysis for musical instrument classification. IEEE Trans Syst Man Cybern Part B (Cybernetics) 38(2):429–438
Diment A, Rajan P, Heittola T, Virtanen T (2013) Modified group delay feature for musical instrument recognition. In: Proceedings of International Symposium of Computer Music Multidisciplinary Research, pp 431–438
Eichner M, Wolff M, Hoffmann R (2006) Instrument classification using hidden Markov models. System 1(2):3
Eronen A, Klapuri A (2000) Musical instrument recognition using cepstral coefficients and temporal features. IEEE Int Conf Acoust Speech Signal Process 2:II753–II756
Essid S, Richard G, David B (2005) Instrument recognition in polyphonic music based on automatic taxonomies. IEEE Trans Audio Speech Lang Process 14(1):68–80
Fourer D, Rouas JL, Hanna P, Robine M (2014) Automatic timbre classification of ethnomusicological audio recordings. In: International Society for Music Information Retrieval Conference (ISMIR 2014)
Fu Z, Lu G, Ting KM, Zhang D (2010) A survey of audio-based music classification and annotation. IEEE Trans Multimed 13(2):303–319
Haidar-Ahmad L (2019) Music and instrument classification using deep learning technics. ReCALL 67(37.00):80–00
Herrera-Boyer P, Peeters G, Dubnov S (2003) Automatic classification of musical instrument sounds. J New Music Res 32(1):3–21
Iowa U (2023) The university of Iowa musical instrument samples publibly available at(mis). https://theremin.music.uiowa.edu/MISPost2012Intro.html
Jordal I (2023) Audiomentaions.https://github.com/iver56/audiomentations
Kingma DP, Ba J (2014) Adam: a method for stochastic optimization. arXiv preprint arXiv:1412.6980
Livshin A, Rodet X (2004) Instrument recognition beyond separate notes-indexing continuous recordings. In: ICMC, pp 1–1
Lostanlen V, Andén J, Lagrange M (2018) Extended playing techniques: the next milestone in musical instrument recognition. In: Proceedings of the 5th International Conference on Digital Libraries for Musicology, pp 1–10
Mahanta SK, Khilji AFUR, Pakray P (2021) Deep neural network for musical instrument recognition using MFCCS. Computación y Sistemas 25:351
Marques J, Moreno PJ (1999) A study of musical instrument classification using gaussian mixture models and support vector machines. Camb Res Lab Tech Rep Ser CRL 4:143
McFee B, Raffel C, Liang D, Ellis DP, McVicar M, Battenberg E, Nieto O (2015) Librosa: audio and music signal analysis in python. Proc Fourteen Python Sci Conf 8:18–25
Oppenheim AV, Schafer RW (2004) From frequency to quefrency: a history of the cepstrum. IEEE Signal Process Mag 21(5):95–106
Park T, Lee T (2015) Musical instrument sound classification with deep convolutional neural network using feature fusion approach. arXiv preprint arXiv:1512.07370
Philharmonia (2023) London philharmonic orchestra dataset publicly. https://philharmonia.co.uk/resources/sound-samples/
Prabavathy S, Rathikarani V, Dhanalakshmi P (2020) Musical instruments classification using pre-trained model. Asian J Electr Sci (AJES) 9(1):45–48
Siebert X, Mélot H, Hulshof C (2023) Study of the robustness of descriptors for musical instruments classification
Singh, P, Bachhav D, Joshi O, Patil N (2019) Implementing musical instrument recognition using CNN and SYM. In: International Research Journal of Engineering and Technology, pp 1487–1493
Solanki A, Pandey S (2019) Music instrument recognition using deep convolutional neural networks. Int J Inf Technol 14:1659–1668
Song Y, Dixon S, Pearce M (2012) A survey of music recommendation systems and future perspectives. Int Symp Comput Music Model Retr 4:395–410
Terasawa H, Slaney M, Berger J (2005) The thirteen colors of timbre. In: IEEE Workshop on Applications of Signal Processing to Audio and Acoustics, IEEE, pp 323–326
Toghiani-Rizi B, Windmark M (2017) Musical instrument recognition using their distinctive characteristics in artificial neural networks. arXiv preprint arXiv:1705.04971
Tu H, Li Y (2023) Neural network for music instrument identification
Valverde-Albacete FJ, Peláez-Moreno C (2014) 100% classification accuracy considered harmful: the normalized information transfer factor explains the accuracy paradox. PLoS ONE 9(1):e84217