Instrument Identification for Real-world Music Pieces

https://github.com/Bai-YT/CS4641-Music-Instrument-Recognition

CS4641 Project Report

Group 6

Yatong Bai
Zhubo Zhou
Tianyi Zuo
Zhaoran Ma
Junzhe Ruan

1. Introduction

1.1 Problem Definition

During an orchestra concert, multiple musical instruments play simultaneously and create an enjoyable experience for the audiences. Unfortunately, not all people can tell the different instruments that musicians are using. To better educate music enthusiasts without a professional background, we improved a machine learning model to auto-recognize the instruments used in a given piece of music, mainly focused on classical music. This study is an interdisciplinary topic between music and machine learning.

1.2 Motivation

Identifying musical instruments has many potential applications. By collecting the instrument's acoustic wave features, we will be able to classify the instruments. Distinguishing musical instruments could help users find specific songs played by a certain instrument, and could also serve as the basis for music generation and other tasks.

This project focuses on the sound-based instrument prediction algorithm and creates a footstone for these more advanced functions. We believe classical music lovers and those interested in learning digital music would benefit from our study.

2. Background

2.1. Existing model

Several existing projects can classify music instruments [1][2][3][4][5]. Among these projects, Renato Profeta's work [1] is the clearest and most well-documented, making it easy to understand and improve.

The model in Renato Profeta's project is based on a small part of the Philharmonia Orchestra Sound Samples music database [6]. It offers a comparison between several machine learning libraries by implementing the same functionality using each of them. The libraries include a support vector machine library (Sklearn), as well as neural network libraries like PyTorch and Keras. After comparing the performance and computational requirements of each library, we chose the Sklearn model for further development.

The model uses Mel frequency cepstral coefficients (MFCC) to extract feature vectors from music pieces and uses Support Vector Machine (SVM) to conduct classification. The model uses cross-validation to evaluate performance. It uses 75% of the data set to train the model and uses the rest 25 % of the data set for testing. As shown in Figure 1, nearly all testing data were correctly classified. This cross-validation result shows that the model works very well for the small segment of the Philharmonia dataset. Compared to the approaches of using CNN, the SVM approach is also fast to train.

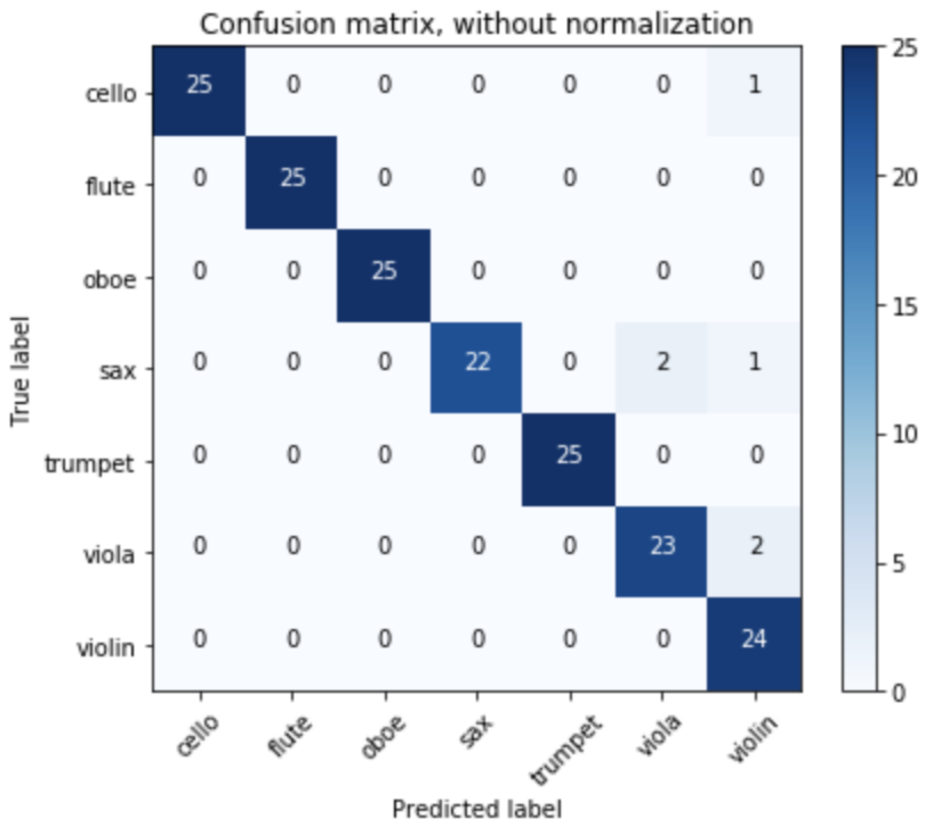


Figure 1. Prediction Results of the Existing Model for Philharmonia Dataset

However, as shown by Figure 2, the model doesn't work so well when predicting a real-word music piece: it identified the instrument of a cello solo piece as viola, and it was not very accurate in predicting violin pieces either. Therefore, we aim to improve the model to allow it to predict real-world music pieces correctly.

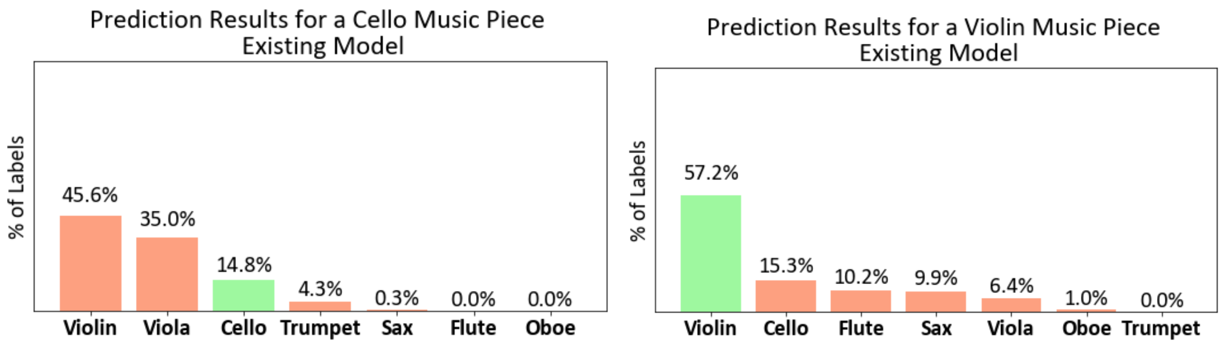


Figure 2. Predation Results for the Existing Model

2.2. Approach

2.2.1. Feature Extraction

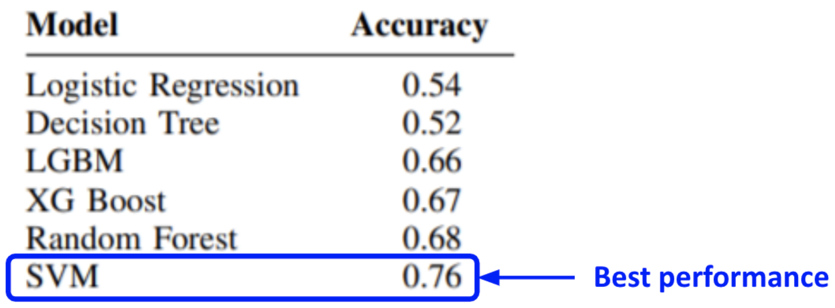
The key to predicting the type of instrument from its sound is to extract feature vectors that represent the timbre information. Some of the common algorithms used to extract timbre information from soundwaves are LPC, MFCC, and SO&CP. Past researchers [2] have experimented with these algorithms, and the results are shown in Table 1. MFCC performs the best as it provides the highest accuracy and gives the perfect result. Therefore, MFCC is chosen as the feature extraction method for this project.

Method	Accuracy
LPC	67%
MFCC	85%
SO&CP	53%

Table 1. Prediction Accuracy of Different Feature Extraction Approaches [2]

2.2.2. Classification

The feature vectors extracted from the soundwave shall be classified using machine learning models. Some common classification models (Logistic regression, Decision tree, SVM, etc) are listed in Table 2, which is from past experiments on instrument-classification algorithms [3]. The experiment results show that among all these methods SVM and CNN perform the best because they have the highest accuracy. However, CNN models are slow to train. Thus, we decided to use the SVM method in our Machine Learning model.



Model	Accuracy
Logistic Regression	0.54
Decision Tree	0.52
LGBM	0.66
XG Boost	0.67
Random Forest	0.68
SVM	0.76

Best performance

Table 2. Prediction Accuracy of Different Classification Approaches [3]

3. Design

3.1. High-level workflow

All dataset files and code are available at our GitHub repository [7].

In this algorithm, Mel-frequency cepstral coefficients (MFCC) method is used to extract feature vectors, while a support vector machine (SVM) is used to classify the feature vectors. The high-level workflow of the algorithm is shown in Figure 3.

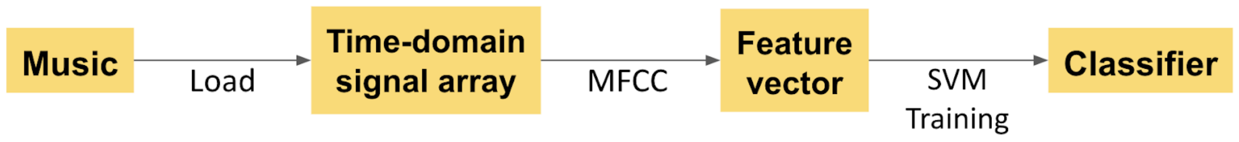


Figure 3. The high-level workflow of the instrument identification algorithm

During training time, each training music file is loaded into an array which stores the time-domain waveform. Then, the MFCC algorithm processes the time-domain array into feature vectors, which are used to train an SVM for classification.

The predicting process is similar. The music snippets to be predicted are loaded as waveforms. MFCC processes them into feature vectors and the SVM makes predictions.

3.2. Feature Engineering (MFCC)

3.2.1. MFCC Overview

MFCC is an algorithm that is capable of characterizing the timbre of the soundwave. Since instrument classification is based on the timbres of the instruments, MFCC perfectly suits this application. MFCC is also widely used in fields like natural language processing.

MFCCs are commonly derived as follows [8]:

1. Take the Fourier transform of a windowed excerpt of a signal.
2. Map the powers of the spectrum obtained above onto the Mel scale, using triangular overlapping windows.
3. Take the logs of the powers at each of the Mel frequencies.
4. Take the discrete cosine transform (DCT) of the list of Mel log powers, as if it were a signal.
5. The MFCCs are the amplitudes of the resulting spectrum.

The output of MFCC is an $n*d$ array, where n is the number of waveform windows, and d is the dimension of the feature vectors. d can be arbitrarily set by the programmer (higher dimension preserves more information). For example, if the length of a waveform file is 1 second and the window size is 0.1 seconds, then n=10. If the dimension of the feature vectors is chosen as 13, then d=13, and the output dimension will be 10*13.

Our algorithm is expected to accept training and testing waveforms with different lengths, which implies different numbers of windows. However, the SVM only accepts data with a fixed predefined dimension. Therefore, in our algorithm, all feature vectors for a particular music file are averaged to characterize the timber of the entire file. To be specific, the $n*d$ output array from MFCC is averaged along the row direction, producing a vector of length d, which contains the timber information of the entire waveform.

3.2.2. Modify the MFCC Parameters to Improve Performance

The MFCC algorithm has several important parameters:

1. The length of the FFT window, which is the length of each signal excerpt.
2. The number of Mel bands. After performing FFT on the time-domain waveform,

the FFT result is assigned into several "Mel bands" or "Mel bins". Increasing the number of Mel bands preserves more information in this step.

3. The dimension of MFCC feature vectors (*d*). After performing DCT on the Mel log powers, the resulting amplitudes are again grouped into several bins, which become the feature vector entries. Increasing the dimension of the feature vectors means increasing the number of bins, which preserves more information in this step.

Preserving more information in the feature vectors should help with the accuracy of the model. When the data size is small, this can result in the risk of overfitting. However, as described in section 3.3.2, the size of the training dataset has been vastly increased. Therefore, even with more MFCC dimensions, the risk for the model to overfit is still low.

To preserve more information, we increased the number of Mel bands and the dimension of output simultaneously. Table 3 summarizes the MFCC parameters of the original and the improved model. Keeping increasing MFCC dimensions might result in even better results, but the required calculation effort of creating feature vectors may also increase. A dimension of 43 balances performance and speed.

	Old Parameters	**New Parameters**
FFT window length	2048	2048
# of mel bands	128	256 (⬆)
Dimension of output	13	43 (⬆)
	Transform files into vectors with length 13	Transform files into vectors with length 43

Table 3. MFCC Parameters of the Original and the Improved Model

3.3. Training Dataset

3.3.1. Philharmonia's dataset overview

The existing model [1], which this project is based on, uses the sound sample dataset created by Philharmonia, a symphony orchestra [6]. The dataset includes 20 types of instruments.

For each instrument, there are about 1000 sound files. Each file records one instrument playing one note at a particular pitch and loudness for a certain duration. Table 4 shows three example combinations of these music properties.

Instrument	**Pitch**	**Duration (sec)**	**Loudness**
Cello	E4	1	Fortissimo (Loud)
Violin	As5	0.5	Fortissimo (Loud)
Sax	F3	1	Pianissimo (Quiet)

Table 4. Properties of the Example Music Files

In this project, we picked 7 of the instruments to build and train the model: cello, violin, viola, trumpet, oboe, sax, flute.

3.3.2. Improvement 1: use the whole Philharmonia dataset

The original model picks 100 files for each instrument from the Philharmonia dataset, which is about one-tenth of the files. Sound files with comparatively high or low pitches and those are very loud or quiet, were removed from the original training data. As a result, the training dataset only consists of a limited range of pitch and loudness diversity.

In the original model, 75% of the dataset was used for training while 25% of the dataset was used for cross-validation. Since both training and validating sets were within the pitch and loudness range, the result of the cross-validation was good. However, other music pieces likely cover a wider range of pitch and loudness. This explains why the model doesn't perform well for real-world music pieces.

To make the model more robust and suitable for the classification of real-world music pieces. The entire Philharmonia dataset was used. By utilizing all available files, the diversity of the dataset was brought back.

3.3.3. Improvement 2: data augmentation

To further augment the diversity of the training dataset to avoid overfitting, data augmentation was performed to the dataset. This was done by overlaying two sound files to create a new waveform. The new data sample is different from either of the two individual sound files, but still can be considered a sound sample played by that instrument.

The overlaid sound file can also simulate the situation in which two instruments of the same type play simultaneously, which is common in music pieces. The overlaying process can be performed by averaging the two waveforms at each sample point.

In this project, for each type of instrument, from the about 1000 sound files in the dataset, about 2000 pairs of recordings were randomly selected and overlaid together.

3.3.4. Improvement 3: use real-world data for training

After implementing the first two improvements, the improvement was limited and the result was still not optimal. The model was marginally better at classifying the instrument type of actual music pieces, but the cross-validation result with the training dataset was superb.

We noticed that all recordings in this dataset were made using one instrument per type. For example, all cello clips are recorded using the same cello, and all violin clips are recorded using the same violin. This might cause incongruity between real-world testing and cross-validation results since different cellos may have different timbres. Having all training data recorded by the same instrument is not ideal since it may cause a large variance. To enhance the diversity in timbre and resolve the issue, we cropped some real-world cello and violin solo pieces into small clips (with the length between ⅓ to 0.5 sec), and added them to the training data.

With the mix of the Philharmonia dataset, the overlaid waveforms, and the real music clips, the total number of training music files are summarized in Table 5:

	Existing Model	**Improved Model**
Cello	100	4506
Violin	100	5410
Flute	100	4297
Oboe	100	2937
Sax	100	3267
Trumpet	100	2332
Viola	100	4797
Total	**700**	**27527**

Table 5. Number of Music Training Files of Different Instruments

4. Results

4.1 MFCC Experiment Result

During the process of improving the model, several MFCC parameter combinations were experimented.

As shown in Figure 5, while experimenting with different MFCC parameters, it was interesting to learn that the prediction accuracy will not improve if only the number of Mel bands or only the number of MFCC features is increased, but the model performance will improve significantly if both parameters are increased simultaneously. The most surprising result is that increasing the number of Mel bands from 128 to 256 while not changing the dimension of MFCC feature vectors worsened the prediction accuracy.

Number of Mel Bands	Number of MFCC Features	Prediction Accuracy
↑	↑	↑
↑	--	↓
--	↑	--

Figure 5. Result after experimenting with different MFCC parameter combinations. "--" means this parameter is unchanged.

4.2 Overall Result (Model Performance)

Using the existing model as a baseline, we started by using a largened Philharmonia Dataset and performed some data augmentation operations. We then modified the MFCC parameters including the length of the FFT window, number of Mel bands, and the dimension of MFCC feature vectors. During the model implementation, we mainly focused on processing more data from the Philharmonia Dataset and increasing the amount of two MFCC parameters (number of Mel bands and dimension of MFCC feature vectors).

For performance evaluation of the models, we used the percentage of correctly predicted snippets of each model as the evaluation criteria. We used a violin solo piece and a cello solo piece for validation. To test our model, we first divided the testing music files into 0.5-sec snippets, and then let the model predict the instrument of each snippet. Since the testing music files are solo pieces (played with only one type of instrument, it is easy to compare the predicted labels to the true labels). We then calculated the percentage of correctly predicted snippets as the evaluation criteria.

We decided to break the relatively long music clips into shorter snippets because this could help us do real-time prediction that can be used in embedded devices. Furthermore, this could be very helpful in situations where instruments alternate to play. By dividing the original music files into snippets, the model could be used to predict the instrument being played at each time instance, and therefore make the analysis result more accurate.

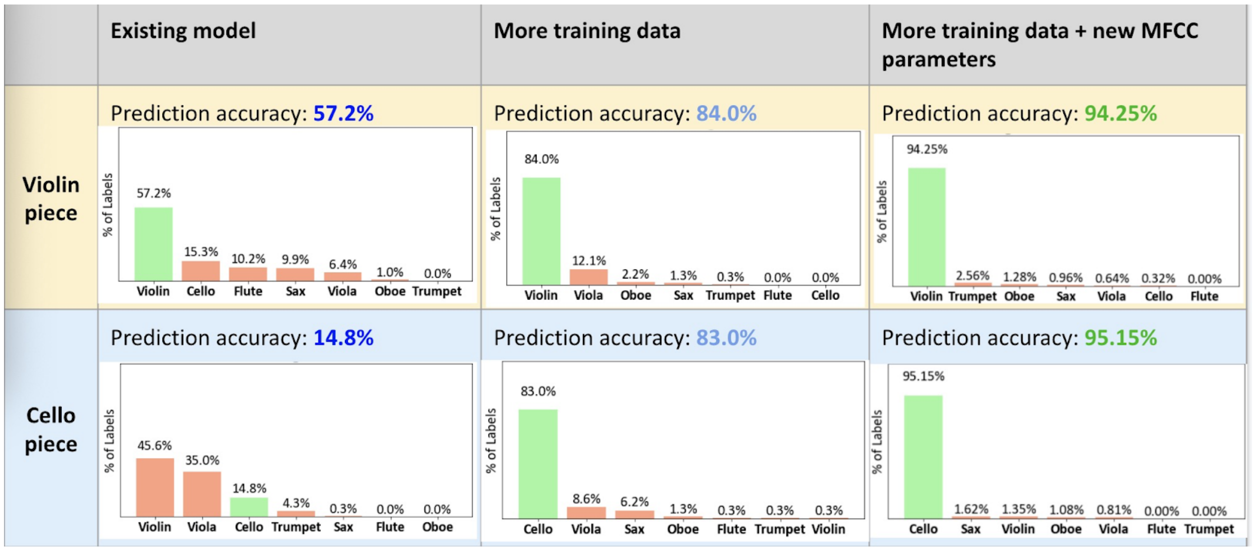


Figure 4. Predation Results for the Existing and the Improved Mode

Figure 4 compares the performance of the improved model with the baseline model. It can be perceived that the prediction is significantly improved with more training data, especially for the cello piece that we could see the accuracy goes from 14.8% to 83%, about a 70% boost in accuracy. Meanwhile, we also had an approximate 30% increase in accuracy for the violin piece. In this way, we improved the accuracy to an average of 83.5% level efficiently.

With the introduction of new MFCC parameters, we were able to further refine the prediction accuracy to the next level. In the end, we have got an average 95% prediction accuracy which would be highly satisfying. We believe with this accuracy, the model could be ready to contribute to applications like education applications.

5. Summary & conclusion

In conclusion, our model successfully used MFCC to extract timbre features and SVM to perform instrument classification. By increasing the diversity of our training set, and tuning the MFCC parameters, we improved the model's prediction accuracy for solo music pieces from an average of 36.0% to an average of 94.7%.

Throughout this project, we learned that adding more training data can make the machine learning model more accurate. Also, it is crucial to have a diverse training set. Lacking diversity will result in poor performance due to overfitting. A good way of ensuring enough diversity is to gather training data from different sources. This is proven by the fact that the model performance was vastly improved after adding real-world music pieces to the training dataset. Furthermore, as the size of the dataset increases, the model can be made more sophisticated, and therefore can allow better exploitation of the training dataset. This is proven by our observation that the model performance is further improved after careful tuning of MFCC parameters.

By applying supervised machine learning techniques to identify instruments, our work could potentially benefit the field by providing an alternative approach for musicians to acquire music scripts. Besides, similar approaches could be taken to separate different soundtracks in music pieces, which can allow producers to have higher freedom in post-processing of music recordings.

References

[1] Guitars, A. I. *BasicsMusicalInstrumClassifi*. Accessed 29 July 2020. github.com/GuitarsAI/BasicsMusicalInstrumClassifi.

[2] Zhao, Zishuo, and Haoyun Wang. “Musical Instrument Classification via Low-Dimensional Feature Vectors.” *Arxiv.org*. N.p., n.d. Web. 29 June 2020.
https://github.com/wiku30/Musical_Instrument_Classification
https://github.com/wiku30/Musical_Instrument_Classification/blob/master/ppt/SST.pptx
https://arxiv.org/pdf/1909.08444.pdf

[3] Racharla, Karthikeya et al. “Predominant Musical Instrument Classification Based on Spectral Features.” n. pag. Web.
https://arxiv.org/pdf/1912.02606.pdf

[4] Das, Orchisama. “Musical Instrument Identification with Supervised Learning.” *Stanford.edu*. N.p., n.d. Web. 29 July 2020.
http://cs229.stanford.edu/proj2019spr/report/6.pdf
http://cs229.stanford.edu/proj2019spr/poster/6.pdf

[5] Agostini, Giulio, et al. “Musical Instrument Timbres Classification with Spectral Features.” *EURASIP Journal on Advances in Signal Processing*, vol. 2003, no. 1, 2003, doi:10.1155/s1110865703210118.
https://link.springer.com/content/pdf/10.1155/S1110865703210118.pdf

[6] “Sound Samples | Philharmonia.” *Philharmonia.co.uk*. N.p., n.d. Web. 5 July 2020.
https://philharmonia.co.uk/resources/sound-samples/

[7] *CS4641-Music-Instrument-Recognition*. Accessed 29 July 2020.
https://github.com/Bai-YT/CS4641-Music-Instrument-Recognition

[8] Sahidullah, and Goutam Saha. “Design, Analysis and Experimental Evaluation of Block Based Transformation in MFCC Computation for Speaker Recognition.” *Speech communication* 54.4 (2012): 543–565. Print.
https://link.springer.com/content/pdf/bbm%3A978-3-319-03116-3%2F1.pdf