The Comparison of Different Feature Extraction Methods in Musical Instrument Classification

N. Rodin*, D. Pinčić*, K. Lenac*, D. Sušanj*
* University of Rijeka, Faculty of Engineering, Rijeka, Croatia
dsusanj@riteh.hr

***Abstract*—In this paper, we analyze four different methods for audio feature extraction and compare their efficiency in the context of musical instrument classification. We study spectrograms, Mel spectrograms, Linear-Frequency Cepstral Coefficients (LFCCs) and Mel-Frequency Cepstral Coefficients (MFCCs) in combination with three different Deep Learning architectures: VGG-16, ResNet-34 and a custom CNN. We investigate the behavior of our models in two different classification scenarios to determine a possible correlation between the number of classes and the efficiency of each method. For this purpose, we took samples from the London Philharmonic Orchestra dataset and ran the experiment for three and fifteen classes of musical instruments belonging to three different instrument families: Woodwinds, Strings and Brass.**

***Keywords*—Musical Instrument Classification, CNN**

I. INTRODUCTION

Music has been a fundamental part of human identity and culture since prehistoric times. Over thousands of years, musical instruments have evolved to a high level of diversity and intricacy, aiming to create unique sounds and trigger specific emotional responses in listeners. From an analytic point of view, the skill of distinguishing between different instruments playing together in a complex musical piece carries an important role. Inspired by this fascinating task, our paper explores different ways of capturing audio features of classical instruments and categorizing them into correct instrument classes. A carefully selected preprocessing method combined with a suitable Deep Learning architecture establishes a foundation for more complex tasks such as music recommendation systems, melody extraction, and music information retrieval. Having that in mind, the goal of this study is to provide an overview of several preprocessing methods and their performance in different classification environments.

In the past years, audio feature extraction and signal processing have become more popular and more researched topics. There have been several studies that tried to compare the efficiency of the existing audio preprocessing methods. J. D. Deng *et al.* [1] analyzed three types of audio features used in musical instrument classification, including perception-based features, MPEG-7 timbral features, and MFCC features. They were compared using different machine learning techniques. In the end, the results showed that MFCC features gave the best classification performance. In addition, A. Eronen [2] compared Linear Prediction Cepstral Coefficients (LPCCs) with MFCCs and concluded that MFCCs gave the best accuracy results when dealing with instrument family classification.

Recently, many audio classification researchers have been using MFCCs as their feature extraction method of choice. This is not only common in studies dealing with musical instrument classification [3], [4], but also fields like speech recognition [5], heart sound classification [6], and animal sound classification [7]. S. K. Mahanta *et al.* [8] combined MFCCs with an artificial neural network and got 97% accuracy on the full London Philharmonic Orchestra dataset. Similar experiment was conducted by S. Prabavathy *et al.* [9] when they used MFCCs in combination with a convolutional neural network to reach the accuracy of 97.5%. Even though MFCCs seem to be more popular, some works such as [10] still opt for Mel spectrograms. Interestingly, R. Profeta and G. Schuller [11] managed to design a new CNN autoencoder-based filter bank that outperformed spectrograms and Mel spectrograms.

Unlike other works, our study encompasses the analysis of four common feature extraction methods that are based on the short-time Fourier transform. Another novelty is the use of multiple CNN architectures and different-sized datasets for comparison.

II. DATASET

For this study, we decided to use a slightly modified version of the London Philharmonic Orchestra dataset [12]. The original dataset consists of 20 instrument classes, including standard orchestral instruments, guitar, mandolin, banjo, saxophone, and various percussion instruments. Each class contains a number of high-quality audio samples, ranging from 74 (banjo) up to 1502 samples (violin).

Considering the significant imbalance of data samples per class, we decided to focus on standard orchestral instruments, thus eliminating the classes with insufficient number of samples. This resulted in a dataset of 15 classes, containing only traditional instruments found in classical orchestral music, with a total of 12 541 audio samples. The distribution of samples in the modified dataset can be observed in Fig. 1.

Seeing that all of the examined instruments belong to three instrument families (Woodwinds, Strings and Brass),

This research was supported by University of Rijeka, Rijeka, Croatia under the grant uniri-tehnic-18-295.

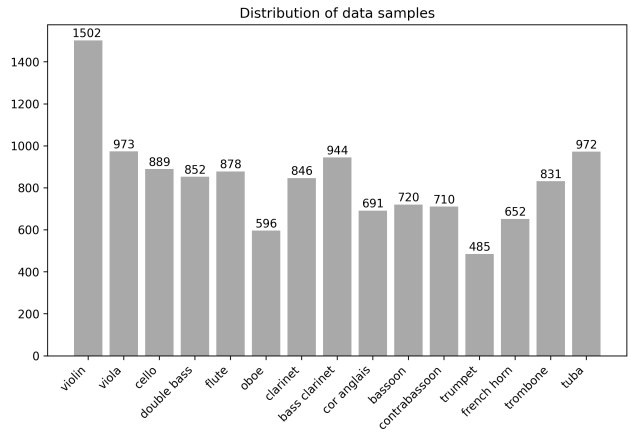
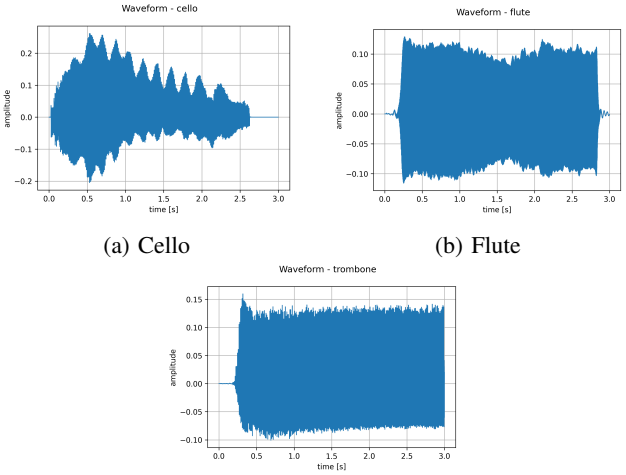


Fig. 1: Distribution of samples in the modified dataset



(a) Cello (b) Flute

(c) Trombone

Fig. 2: Waveforms of single notes being played on different instruments

we decided to create an additional experiment with a reduced dataset containing only three classes of choice, each representing one instrument family. For this purpose, we opted for cello, flute, and trombone, as they appeared to have a reasonably balanced number of samples. The total number of samples in the smaller dataset was 2 598.

The audio samples consisted of single tones and simple phrases played on different musical notes ranging from A1 to G7. In addition, samples were played with a wide range of techniques and dynamics.

The duration of audio files varied from less than a second to over a minute, which meant we had to decide upon a fixed length and perform trimming or padding, depending on the sample. The chosen duration was 3 seconds, as proposed by [8], stating that all of the 'onsets and attacks of the sound take place within the first 3 seconds'. To facilitate data handling, we converted all data samples from the original MP3 audio file format to WAV format. Additionally, we generated a CSV file containing file names, folder names, and corresponding class labels for each audio sample to make annotation easier during preprocessing and training.

III. METHODS

A. *Preprocessing*

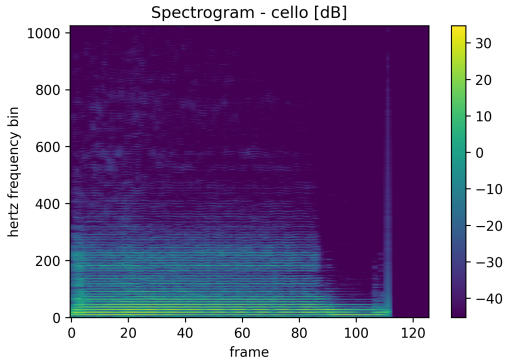
When it comes to processing and feeding audio data into neural networks, several concepts need to be considered. The sound produced by a musical instrument is an oscillation of air molecules caused by the vibration of the sound source (e.g., strings, reeds, or lips). This periodic alternation of high and low air pressure determines a sound wave, which can be represented by a waveform (Fig. 2). A waveform is a visual representation of the raw audio signal in the time domain, having amplitude values on the y-axis, and time on the x-axis.

In the real world, audio signals are rarely just pure sine or cosine waves. Nevertheless, it is possible to approximate complex periodic signals using a sum of elementary trigonometric functions of different wavelengths, frequencies, and amplitudes. This is known as the Fourier series. We heavily rely on this concept during sound analysis when we try to extract information about the signal's constituent parts to gain a better understanding of its nature and origin.

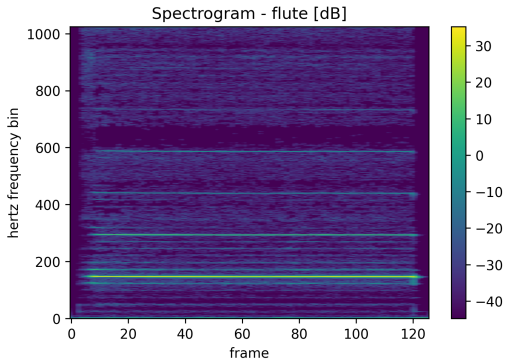
In the following text, we describe four sound preprocessing methods, each focusing on different components of audio signals and human perception of sound.

Spectrograms. One of the most observed wave features of audio signals is their frequency. Knowing that complex sound is composed of many sinusoids with different frequencies, a lot can be learned by extracting a frequency-power spectrum of the signal. A spectrum represents a distribution of frequencies over the entire time domain of the signal. It is performed by the fast Fourier transform (FFT), and it gives information about how much each frequency component (aka harmonic) contributes to the overall waveform of the signal. It is generally presented as a two-dimensional graph in the frequency domain, having magnitude on the y-axis and frequencies measured in hertz on the x-axis. However, considering that this type of representation captures the frequency distribution of the entire waveform, the information about frequencies at each given moment in time is lost.

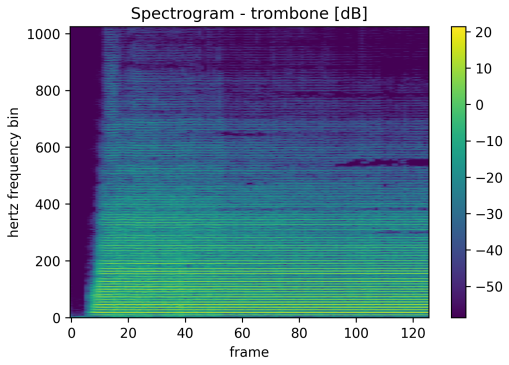
To preserve time domain features, a more profound transform is applied to the raw signal. This is called a short-time Fourier transform (STFT). STFT consists of many FFTs which are computed throughout several time intervals. These time intervals are called frames, and their lengths are defined by a frame size measured in samples. Another important parameter is the hop length, which determines the distance in samples between each frame. Setting a hop length that is smaller than the frame size causes frames to overlap, thus minimizing spectral leakage at the edges of the transformed signal. Finally, as a result of STFT, we get a spectrogram. Spectrograms are three-dimensional representations of sound in both frequency and time domains, with frames on the x-axis, frequency on



(a) Cello



(b) Flute



(c) Trombone

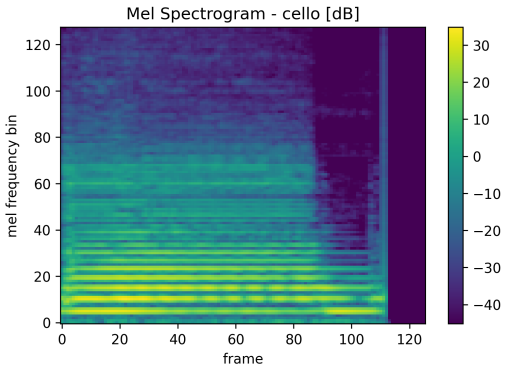
Fig. 3: Spectrograms

the y-axis, and magnitude on the z-axis, often visualized by a color bar.

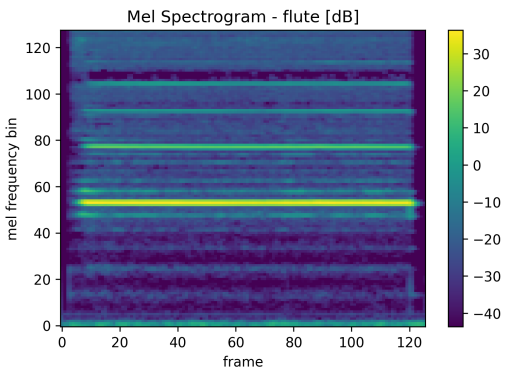
Fig. 3 shows spectrograms of a single note being played by cello, flute, and trombone. In our case, a sample rate of 22.05 kHz was used for all audio samples. To perform STFT, we used a frame size of 2048 samples and a hop length of 512. This resulted in a 1025×126 sized matrix which was going to be treated as input data for our classification models. The y-axis shows frequency bins instead of continuous frequency. Frequency bins are simply intervals of continuous frequency whose width can be calculated as the sample rate divided by the frame size of the FFT. In our case, having a sample rate of 22.05 kHz, and a frame size of 2048, each bin represents around 10.77 Hz in the frequency domain.

Another point worth emphasizing is the measurement unit on the magnitude axis, which has to do with the fact that the human auditory system is logarithmic in nature. This means that we perceive certain features of sound, such as pitch and loudness, in a nonlinear fashion. For this reason, we decided to represent magnitude in decibels (dB), which is a measure of intensity level based on the logarithmic scale. This was used throughout all preprocessing methods, in an attempt to mimic the way humans perceive sound.

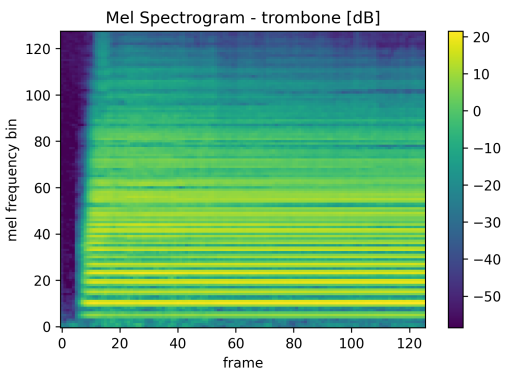
Mel spectrograms. Another important characteristic of sound perception is pitch. Despite being closely correlated to frequency, pitch is a subjective measure that portrays nonlinear tendencies. Because of this, the same differences in the lower frequency range are perceived as further apart



(a) Cello



(b) Flute



(c) Trombone

Fig. 4: Mel spectrograms

than those in the higher frequency range. To create a perceptually relevant frequency representation of sound, we apply the logarithmic scale on the frequency values of the spectrogram, which results in a Mel spectrogram. Equal distances between values on the mel scale have the same perceptual distance in pitch, with 1000 mels being equal to 1000 Hz, by agreement. When creating a Mel spectrogram, it is important to define the number of mel bands to be used during conversion. Mel bands define the range of perceptually relevant frequencies. More specifically, they determine which points of the signal's frequency range will be converted into mel scale. In our case, 128 mel bands were used, which ultimately resulted in data samples of size 128×126.

Fig. 4 shows Mel spectrograms of cello, flute, and trombone. Equation (1) is used to convert frequency expressed in hertz to frequency expressed in mels.

$$m = 2595 \cdot \log(1 + \frac{f}{700}) \tag{1}$$

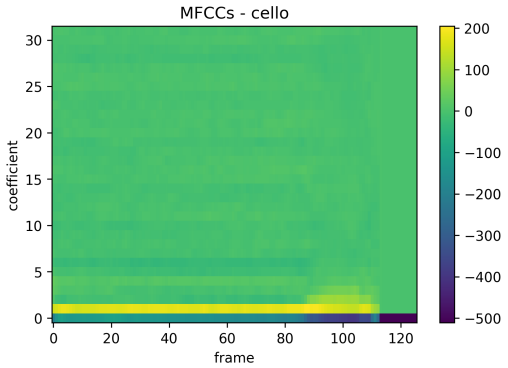
MFCCs. Another highly subjective aspect of sound is timbre. It is often described as the *color* of sound that is unique to each and every instrument. Timbre is what acoustically sets apart two instruments playing the same note at the same intensity, for the same duration of time.

Features that are commonly used to extract timbral and textural aspects of sound are Mel-Frequency Cepstral Coefficients or MFCCs. They convey information about the spectral envelope, spectral detail, harmonic content, and amplitude and frequency modulation of the signal (aka *tremolo* and *vibrato*). Some of these elements can easily be mapped onto characteristics of speech generation, specifically glottal pulse, formants, and frequency response of the vocal tract. For this reason, MFCCs are often used in speech recognition tasks [5].

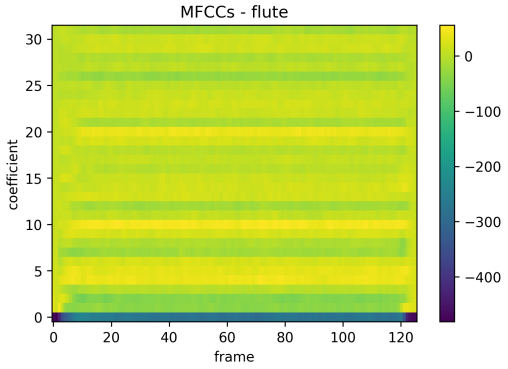
The signal's cepstrum is computed as an inverse Fourier transform applied on the log amplitude spectrum of the signal. This is given by the expression in (2). Cepstral coefficients with positive values signalize that spectral energy is mostly present in the lower frequency regions, whereas negative values indicate higher frequencies.

$$C[x(t)] = F^{-1}(\log(F[x(t)]) \tag{2}$$

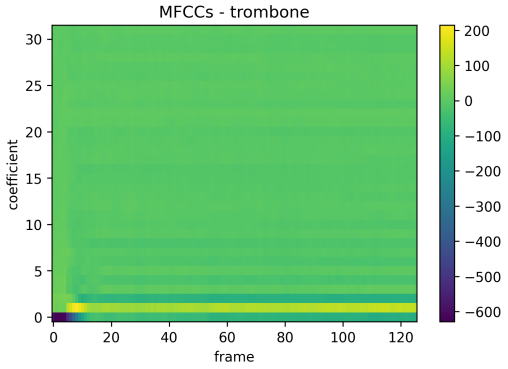
When extracting audio features using the MFCC method, several parameters have to be defined. To stay consistent with the rest of examined methods, we used 22.05 kHz as the sampling rate, 2048 samples for the frame size, the hop length of 512 samples, and 128 mel bands. Due to the input size restriction of our VGG classification model, 32 MFCC coefficients were chosen instead of 13, which is a number traditionally used in machine learning. The reason behind using the lowest possible value of MFCCs is the fact that they contain the most relevant timbral information, specifically the characteristics of the spectral envelope. As a result of preprocessing using the MFCC method, we get data points of size 32×126. A visual representation of MFCCs for cello, flute and trombone can be seen in Fig. 5.



(a) Cello



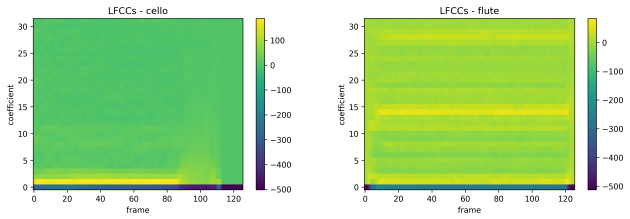
(b) Flute



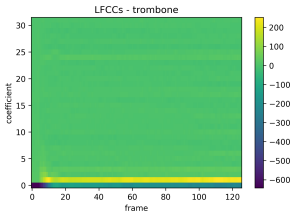
(c) Trombone

Fig. 5: MFCCs

LFCCs. Linear-Frequency Cepstral Coefficients, or LFCCs, are a feature extraction method similar to MFCCs. The only major difference is that they use a linear frequency scale instead of the logarithmic one. This way, audio signals are being treated in disregard to human perception of pitch, therefore preserving a better resolution in the higher frequency regions of the sound [13]. To perform preprocessing, we used 32 LFCC coefficients, due to the previously mentioned reasons. Since LFCCs don't use mel scaling, there was no need to define the number of mel bands. All of the other parameters remained the same as in previous methods. The resulting matrix had the same dimensions as the one derived by the MFCC method, i.e.,



(a) Cello (b) Flute



(c) Trombone

Fig. 6: LFCCs

32×126. LFCCs of a single note played by cello, flute and trombone are visualized in Fig. 6.

B. Classification

To get a better understanding of how each preprocessing method contributes to the overall efficiency of instrument classification, we decided to compare them using three different Deep Learning architectures.

The first architecture was a simple, custom-made convolutional neural network with four convolutional blocks, one flatten layer, one linear layer and a softmax layer at the end. Each convolutional block contained one 2D convolutional layer, a ReLu activation function, and a 2D max pooling layer. The second architecture was the standard VGG-16 convolutional neural network which consists of 16 convolutional layers. The first Visual Geometry Group (VGG) architecture was proposed by [14]. Lastly, we used ResNet-34 as our third architecture. ResNet-34 is a state-of-the-art image classification model that consists of 34 convolutional layers, and was first presented in [15].

The reason behind using several Deep Learning architectures in our experiments was to compare the preprocessing methods with a more general approach, as well as to eliminate any potential biases.

Furthermore, the dataset was split randomly into 80% of training samples and 20% of test samples. All of the methods were trained and tested on the same data samples. The models were trained for 30 epochs, using a batch size of 128. We used the cross-entropy loss function and the Adam optimizer with parameters Beta1 = 0.9 and Beta2 = 0.999. A learning rate of 0.00001 was used for all methods and models, and weight decay was set to None. The models were evaluated after the last epoch using test samples. Additionally, the experiment was repeated several times for each method and the best accuracy results were saved for analysis.

IV. RESULTS

The accuracies of the trained models for the smaller and larger dataset are shown in Table I and Table II, respectively. The tables display the best-achieved accuracy results for each method.

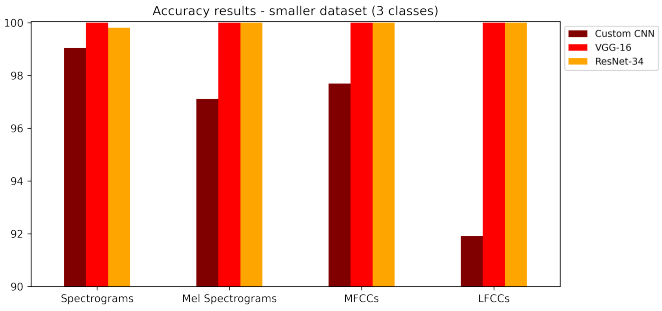
TABLE I: Results - smaller dataset (3 classes)

	Custom CNN	VGG-16	ResNet-34
Spectrograms	99.04%	100.0%	99.81%
Mel spectrograms	97.11%	100.0%	100.0%
MFCCs	97.69%	100.0%	100.0%
LFCCs	91.91%	100.0%	100.0%

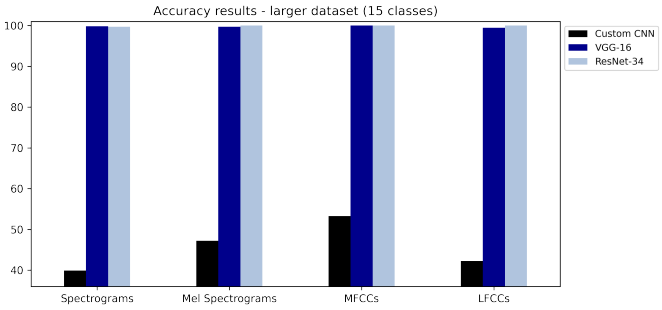
TABLE II: Results - larger dataset (15 classes)

	Custom CNN	VGG-16	ResNet-34
Spectrograms	39.91%	99.84%	99.72%
Mel spectrograms	47.17%	99.72%	100.0%
MFCCs	53.27%	100.0%	100.0%
LFCCs	42.22%	99.44%	100.0%

When observing the results for the smaller dataset, we notice an interesting occurrence of 100% accuracy throughout majority of the cases. This becomes particularly apparent with more complex Deep Learning architectures (VGG and ResNet). In this case, all 3 classes were noticeably distinct, which resulted in deep learning architectures being able to successfully classify the instruments in all cases, thanks to their high generalization ability, especially on small dataset with distinct samples. The dataset contains simple musical notes and phrases played similarly within each class, resulting in low intra-class variance. This allowed models to focus on inter-class



(a) Smaller dataset (3 classes)



(b) Larger dataset (15 classes)

Fig. 7: Visual representation of accuracy results

differentiation, leading to effective performance across all methods.

Judging by the accuracy of the custom CNN, spectrograms seem to have given the best results when it came to the smaller dataset. They were followed by MFCCs and Mel spectrograms, while LFCCs produced noticeably lower results. However, it is important to note that during training, spectrograms required a lot more memory and computational power than other methods. This may be due to a significant difference in input size that we get as a result of preprocessing with spectrograms.

If we look at the larger dataset, the efficiency of each method becomes more apparent. Although comparing the methods in combination with VGG and ResNet doesn't provide us with much information, the results of the custom CNN display clear differences - MFCCs seem to outperform other methods, with Mel spectrograms taking the second place. LFCCs along with spectrograms produce poor results when tested on a dataset with less distinguishable classes.

These results can be explained through the nature of each feature extraction method. Both MFCCs and Mel spectrograms use the mel scale instead of linear scale to preserve the perceptual relevance of the frequency spectrum. MFCCs, being built on top of Mel spectrograms, extract additional information about timbral and textural aspects of sound. Therefore, seeing that our dataset consists of instruments playing the same notes with similar intensities, timbre becomes the most important mean of distinguishing between closely related classes. This is why MFCCs tend to outperform other feature extraction methods in the task of musical instrument classification. MFCCs combined with strong ResNet or VGG architectures serve as an excellent cornerstone for building successful audio classification models.

V. CONCLUSION

The goal of this paper was to provide an overview of four preprocessing methods and their efficiency in the task of musical instrument classification. We ran several experiments and compared spectrograms, Mel spectrograms, MFCCs and LFCCs as tools for audio feature extraction. The preprocessed data was fed into three types of Deep Learning architectures, and the experiments were repeated for three and fifteen classes of musical instruments. Spectrograms showed an unusually high performance when it came to a simpler dataset. However, as the number of classes grew higher, their performance significantly dropped. On the other hand, the rest of the preprocessing methods portrayed consistency in both scenarios. The results showed that MFCCs performed the best job of capturing crucial audio features, especially when it came to less distinguishable classes.

In conclusion, a thoughtful selection of the preprocessing method can significantly impact the results of classification. For this reason, it is important to understand which features of sound are relevant to the given problem. Although a strong Deep Learning architecture can partially alleviate the effects of preprocessing, it is still important to pay attention to the nature of each feature extraction method.

REFERENCES

[1] J. D. Deng, C. Simmermacher, and S. Cranefield, "A study on feature analysis for musical instrument classification," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 38, no. 2, pp. 429–438, 2008.

[2] A. Eronen, "Comparison of features for musical instrument recognition," in *Proceedings of the 2001 IEEE Workshop on the Applications of Signal Processing to Audio and Acoustics (Cat. No.01TH8575)*, 2001, pp. 19–22.

[3] S. Chakraborty and R. Parekh, *Improved Musical Instrument Classification Using Cepstral Coefficients and Neural Networks*, 09 2018, pp. 123–138.

[4] K. Racharla, V. Kumar, C. B. Jayant, A. Khairkar, and P. Harish, "Predominant musical instrument classification based on spectral features," in *2020 7th International Conference on Signal Processing and Integrated Networks (SPIN)*, 2020, pp. 617–622.

[5] W. Han, C.-F. Chan, C.-S. Choy, and K.-P. Pun, "An efficient mfcc extraction method in speech recognition," in *2006 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2006, p. 4.

[6] M. Deng, T. Meng, J. Cao, S. Wang, J. Zhang, and H. Fan, "Heart sound classification based on improved mfcc features and convolutional recurrent neural networks," *Neural Networks*, vol. 130, pp. 22–32, 2020.

[7] E. Şaşmaz and F. B. Tek, "Animal sound classification using a convolutional neural network," in *2018 3rd International Conference on Computer Science and Engineering (UBMK)*, 2018, pp. 625–629.

[8] S. K. Mahanta, A. F. U. R. Khilji, and P. Pakray, "Deep neural network for musical instrument recognition using mfccs," *Computación y Sistemas*, vol. 25, no. 2, pp. 351–360, 2021.

[9] S. Prabavathy, V. Rathikarani, and P. Dhanalakshmi, "An enhanced musical instrument classification using deep convolutional neural network," *International Journal of Recent Technology and Engineering (IJRTE)*, vol. 8, 11 2019.

[10] D. Szeliga, P. Tarasiuk, B. Stasiak, and P. S. Szczepaniak, "Musical instrument recognition with a convolutional neural network and staged training," *Procedia Computer Science*, vol. 207, pp. 2493–2502, 2022, knowledge-Based and Intelligent Information Engineering Systems: Proceedings of the 26th International Conference KES2022.

[11] R. Profeta and G. Schuller, "End-to-end learning for musical instruments classification," in *2021 55th Asilomar Conference on Signals, Systems, and Computers*, 2021, pp. 1607–1611.

[12] London philharmonic orchestra dataset. [Online]. Available: https://philharmonia.co.uk/resources/sound-samples/

[13] X. Zhou, D. Garcia-Romero, R. Duraiswami, C. Espy-Wilson, and S. Shamma, "Linear versus mel frequency cepstral coefficients for speaker recognition," in *2011 IEEE Workshop on Automatic Speech Recognition Understanding*, 2011, pp. 559–564.

[14] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014.

[15] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.