timbres produced on different areas of the guitar. The timbres selected are the following:

1) Palm on lower guitar body
2) Fingers on lower side
3) Thumb on top guitar body
4) Fingers on the keyboard (muted strings)

All the experiments were performed using a training dataset of 390 audio samples (~100 per timbre) and a test dataset composed of 80 samples. All of the recordings were performed on a Yamaha APX-8A acoustic guitar using the internal piezoelectric transducers and pre-amplifier. Note this standard setup is not ideal for all kind of sounds; as a matter of fact more modern amplification systems integrate a condenser microphone and/or a magnetic pickup.

Two studies (`study-1`, `study-2`) were carried out to compare the performance of the original K-nearest-neighbors classifier from timbreID and a feed forward neural network.

The configuration for `study-1` uses the Bark onset detector chained to the BFCC feature extractor and finally to the Knn classifier (timbreID object), k = 3.

The configuration for `study-2` is identical except for the classifier that is now replaced by a feed forward neural network, composed by an input layer of 25 neurons, 2 hidden layers of 25 neurons each and an output layer of 4 neurons. Sparse Categorical Crossentropy is the loss function of choice and the optimizer used was Adam. The number of training epochs ranges from 150 to 600 epochs depending on the test case. For both studies the sample rate is 48000Hz, Bark and Bfcc use a window size of 1024 samples and spacing of 0.5 barks. This produces a vector of 50 coefficients that is reduced by retaining only the first 25 values. The hop size for Bark is 128 samples.

Both experiments were repeated by introducing a delay between the onset detection and the feature extraction, in order to capture a different part of each sound. This should reduce pre-onset resonance and general noise in the extracted values. Given that ”the time between an instrument onset and its attack peak varies unpredictably according to the instrument” [8], different delay values have been tested: Bark already introduces a delay of at least 5.33ms (sample rate: 48000Hz) and the 2 values tested for the total delay were 20ms and 10ms, meaning respectively additional delays of 14.66ms and 4.66ms. Given the results, an additional test was run with the `study-1` configuration, without reducing the feature vector.

III. Results

TABLE I. `study-1` scores (Knn)

A-Delay	0ms	4.66ms	14.66ms
Study-1 Accuracy:	0.8125	0.8250	**0.4625**
Study-2 Accuracy:	**0.9250**	**0.9000**	0.4125

From the results in tables I the Neural Network architecture shows to be the clear winner in terms of performance: this is in all likelihood due to the ability of a network of neurons to learn weights for the input features, while K-nearest-neighbors is limited to equal weighting or human designed weights.

Studies using *20ms* total delay showed a great reduction in accuracy while *10ms* results were closer to the score of the setup without delay. More tests need to be done with this parameter, in particular in the real application with a guitar connected to the recognition system: this should prove to work better in that case, even with similar accuracy in the test, because rapid successions of different sounds cause interference between said sounds.

This pilot study used only the *bfcc* feature extractor as previous studies showed that it is more fitting for percussive sounds than other modules offered by the library [8], but more experiments will be carried out with more extractors. The additional test carried out on a smaller dataset with the full 50 values vector computed by *bfcc* was evaluated with a Principal Component Analysis plot (Fig. 1) and compared to the PCA plot of the dataset for the other studies: the greater number of feature show marginally better separation. This, along other configurations will be tested.

Every configuration was successfully compiled as a headless plugin for Elk Audio OS and tested with a guitar.

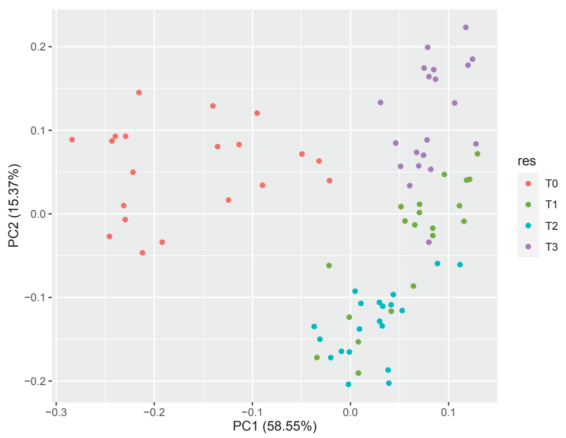

Fig. 1. PCA, 50 features, 20 samples per class. T0:Palm on lower body, T1:Fingers on lower side, T2:Thumb on top body, T3:Fingers on keyboard

References

[1] L. Turchet, “Smart Musical Instruments: vision, design principles, and future directions,” *IEEE Access*, vol. 7, pp. 8944–8963, 2019.
[2] L. Turchet, M. Benincaso, and C. Fischione, “Examples of use cases with smart instruments,” in *Proceedings of Audio Mostly Conference*, 2017, pp. 47:1–47:5.
[3] L. Turchet, A. McPherson, and M. Barthet, “Real-time hit classification in a Smart Cajón,” *Frontiers in ICT*, vol. 5, no. 16, 2018.
[4] E. Meneses, J. Wang, S. Freire, and M. M. Wanderley, “A comparison of open-source linux frameworks for an augmented musical instrument implementation,” in *Proceedings of the Conference on New Interfaces for Musical Expression*, 2019, pp. 222–227.
[5] J. Burgoyne, I. Fujinaga, and J. Downie, “Music information retrieval,” *A New Companion to Digital Humanities*, pp. 213–228, 2016.
[6] W. Brent, “A timbre analysis and classification toolkit for pure data,” in *Proceedings of the International Computer Music Conference*, 2010.
[7] A. Martelloni, A. P. McPherson, and M. Barthet, “Percussive fingerstyle guitar through the lens of nime: an interview study,” 2020.
[8] W. Brent, “Cepstral analysis tools for percussive timbre identification,” in *Proceedings of the International Pure Data Convention*, 2009.