Stefani and Turchet
EURASIP Journal on Audio, Speech, and Music Processing (2025) 2025:28
https://doi.org/10.1186/s13636-025-00413-6

EURASIP Journal on Audio, Speech, and Music Processing

EMPIRICAL RESEARCH

Open Access

Real-time playing technique recognition embedded in a smart acoustic guitar

Domenico Stefani[1*] and Luca Turchet[1]

Abstract

The integration of real-time music information retrieval techniques into musical instruments is a crucial step towards smart musical instruments that can reason about the musical context. This paper presents a real-time guitar playing technique recognition system for a smart electro-acoustic guitar. The proposed system comprises a software recognition pipeline running on a Raspberry Pi 4 and is designed to listen to the guitar's audio signal and classify each note into eight playing techniques, both pitched and percussive. Real-time playing technique information is used in real-time to allow the musician to control wirelessly-connected stage equipment during performance. The recognition pipeline includes an onset detector, feature extractors, and a convolutional neural classifier. Four pipeline configurations are proposed, striking different balances between accuracy and sound-to-result latency. Results show how optimal performance improvements occur when latency constraints are increased from 15 to 45 ms, with performance varying between pitched and percussive techniques based on available audio context. Our findings highlight the challenges of generalization across players and instruments, demonstrating that accurate recognition requires substantial datasets and carefully selected cross-validation strategies. The research also reveals how individual player styles significantly impact technique recognition performance.

Keywords Music information retrieval, Embedded audio, Real-time signal processing, Playing techniques

1 Introduction

The convergence of the Internet of Things [1] with sound and music computing [2] has fostered the emergence of computing and networking paradigms such as the Internet of Musical Things (IoMusT) [3], which offers to bridge the technological and creative potentials of networked sound and musical devices. IoMusT refers to networks of interconnected musical devices that can sense, process, and exchange sound and music-related data, enabling new forms of musical interaction and collaboration. These devices include Smart Musical Instruments (SMIs) [4] and other music-related objects [5, 6]. SMIs represent a key component of the IoMusT and have been defined as a family of Digital Musical Instruments (DMIs) characterized by their self-contained nature and the use of sensors, actuators, *embedded intelligence*, as well as network connectivity that allows these to become "Musical Things" [7].

Music information retrieval (MIR) is the research field that investigates the retrieval of musical features and music-related information. MIR algorithms are a key feature for SMIs, as they are instrumental to the constitution of embedded intelligence by enabling "reasoning" about the musical context. In particular, real-time Music information retrieval (RT-MIR) algorithms extract musical properties from musical signals as they are generated by the musician. Such information can then be immediately repurposed to trigger, affect, or control in-device processes such as sound synthesis. Alternatively, it can be shared with a network of other SMIs and Musical Things. Wirelessly shared musical properties can be used to control elements of a performance such as stage-lighting effects, video effects,

*Correspondence:
Domenico Stefani
domenico.stefani@unitn.it
[1] Department of Information Engineering and Computer Science, University of Trento, Via Sommarive, 9, 38123 Trento, Italy

Springer Open

© The Author(s) 2025. **Open Access** This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/.

or virtual effects in extended-reality performances [8]. Of crucial importance for RT-MIR systems—along with their accuracy—is their *latency*, which refers to the delay between the input signal and when the relevant properties ("results") are extracted. RT-MIR methods are under more stringent requirements than offline MIR, as their computation time and other delays, as well as precision and accuracy, are under scrutiny. Moreover, computation time is frequently determined by the capabilities of the computing device executing the algorithm.

In this paper, we contribute to this growing domain with the development of a real-time playing technique recognition system embedded within a smart acoustic guitar prototype (see Figs. 1 and 2). During a performance, the recognition system detects notes in the audio signal and classifies them according to its knowledge of eight distinct playing techniques, among pitched and percussive techniques on the acoustic guitar. Four different configurations of the recognition system are analyzed, each corresponding to a different accuracy-latency tradeoff. The recognition is performed on a small resource-constrained single-board computer (SBC) that can be embedded into the instrument. The smart guitar prototype is integrated into a proof-of-concept IoMusT environment, with in-device sound synthesis alongside wirelessly controlled projections and stage effects.

We conduct three main experiments to get a deeper understanding of the system by assessing its performance in different scenarios. For Experiment 1 we set to evaluate accuracies and latencies achieved by the proposed recognition system with different size of the context window. For Experiment 2, we set to evaluate the generalization performance over multiple guitars and players. Moreover, we set to verify whether the current task is subject to confounding factors [9] akin to the artist and album effects [10, 11] identified in other music classification tasks. These effects refer to the artificial inflation of classification performances caused by validation strategies that allow the same artists or albums in both

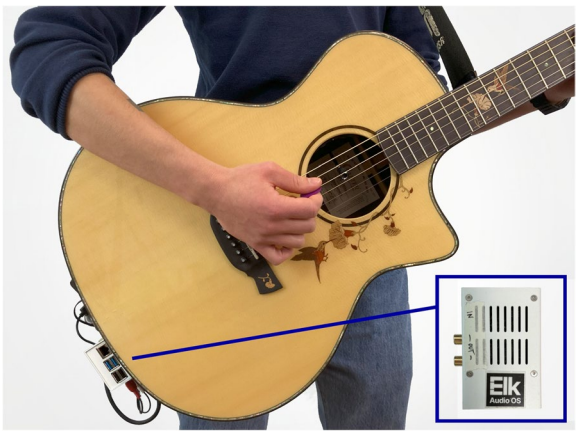

Fig. 1 Prototype of smart guitar connected to the playing technique recognition system. On the lower left, the image shows a Raspberry Pi 4 with a HiFiBerry audio hat and enclosure attached to the instrument. The embedded computer receives power from a USB power bank and the audio signal from the guitar's transducers, which include a piezoelectric pickup and an internal condenser microphone

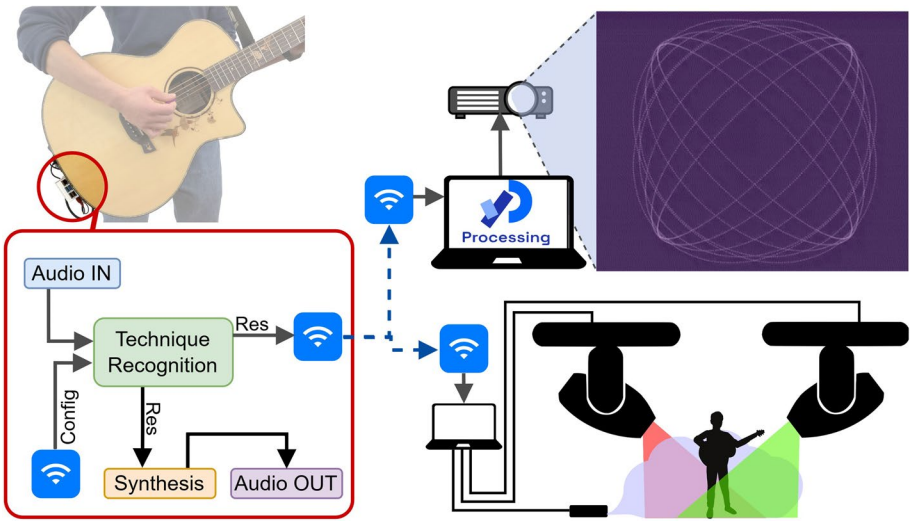

Fig. 2 Proof-of-concept IoMusT environment in which the proposed system was integrated. On the left, the image shows the processing performed inside the SBC, which includes wireless control (e.g., start/stop), playing technique recognition, wireless communication of the results, and internal sound synthesis influenced by the recognition results. The predicted technique was shared through Open Sound Control (OSC) messages in a local network and used to control projected visuals and stage effects (i.e., moving-head lights and a smoke machine)

training and testing recordings. For our task, performer-instrument pairs in the dataset are natural data groups that could represent a confounding factor if not properly separated. We refer to the potential effect as performer-instrument effect and set out to assess whether it has an impact on the current task. Finally, for Experiment 3, we evaluate the performance when specializing to one guitar and the impact of individuality in the playing style of different performers [12] on recognition.

The three experiments aim to evaluate whether intuitively existing effects (performer-instrument effect, performer individuality) impact a timbre-based approach to real-time playing technique recognition, particularly with very short signal context.

The remainder of the paper is organized as follows. In Section 2, we introduce the background and review related works on guitar augmentations, IoMusT, guitar technique recognition, and percussive fingerstyle recognition. Section 3 describes the methodology, including the hardware, and software used. Section 4 presents the dataset used for evaluation and three different experiments conducted. In Section 5 we present and discuss the results of the experiments. Finally, we draw our conclusions in Section 6.

2 Background

2.1 Guitar technology and augmentations

In the last 50 years, the evolution of the guitar as an instrument has benefited from advancements in the field of electronics and computer science. In particular, digital technology enabled the use of computing devices for advanced signal processing and fostered the creation of many DMIs. This made it possible to extend the creative possibilities of several instruments, including the guitar, through sensors, actuators, or simply clever signal-processing techniques. These have also been referred to as Augmented Musical Instruments (AMIs), and augmented guitars respectively, in the field of New Interfaces for Musical Expression (NIME) [13].

Augmenting a musical instrument often involves the use of a computing unit to process audio or sensor signals, and this can be either an external device (e.g., laptop) or an embedded device. Due to the computing power constraints of embedded devices, PCs have been widely used to implement non-embedded solutions for AMIs. In the case of augmented guitars, projects that involved the use of PCs as computing units include Puckette's SMECK guitar [14], the Mobile Wireless Augmented Guitar [15], the Multimodal Guitar [16], the guitar performance visualizer by Angulo et al. [17], Lähdeoja's augmented guitar [18], and both GuiaRT and GuitarAMI [19]. Despite the quick prototyping capabilities of PC-based augmentations, the use of a separate computing unit makes for cumbersome performance setups with laptops, audio interfaces, and either inconvenient cables or wireless connectivity with compromises on audio quality [15]. Other closed-source/commercial solutions include many MIDI guitars, the Fishman Triple Play[1] pickup, Roland GR/SY systems[2], and AXON USB systems[3], which still rely on a separate processing unit such as a computer, synthesizer, or pedalboard.

Examples of embedded guitar augmentations are instead provided by instruments such as the Line 6 Variax[4], Fender VG[5], and Gibson Firebird X[6]. These offered the possibility of emulating different stringed instruments or adding effects through internal audio processing, freeing up the guitarist from the need for additional hardware and complex setups. These are, however, often closed systems that are limited to predefined sets of modeling algorithms and effects. A more in-depth review of guitar augmentation technologies is beyond the scope of this paper and can be found in [20].

Some of the core attributes of AMIs are shared with the concept of Smart Musical Instruments, which is described in the next section.

2.2 IoS, IoMusT, and smart musical instruments

The concept of IoMusT was introduced in [3] as an extension of the concept of Internet of Things (IoT) focused on the technological infrastructure that enables the creation of ecosystems of interconnected devices that allow interactions between musicians and audiences.

A prominent example of a musical thing in the IoMusT is that of SMIs [7], a family of interoperable musical instruments equipped with sensing and actuation capabilities, along with embedded reasoning abilities and wireless interconnecting technology. One of the early examples in this class of instruments was the Sensus smart guitar by Elk Audio[7], which was equipped with various sensors, physical controls, wireless connectivity devices, and actuators. Sensus processed the audio signal internally and it used wireless network connections for both audio and control signals. Other relevant examples of smart guitars are those developed by HyVibe[8] and Lava Music[9].

[1] https://www.fishman.com/tripleplay/
[2] https://www.roland.com/it/products/gr-55/
[3] https://www.soundonsound.com/reviews/terratec-axon-ax50-usb
[4] https://line6.com/variax-modeling-guitars/
[5] https://www.roland.com/products/g-5/
[6] https://legacy.gibson.com/Products/Electric-Guitars/Firebird/Gibson-USA/Firebird-X/Specs.aspx
[7] https://www.youtube.com/watch?v=fqzEQnsSIoY
[8] https://www.hyvibeguitar.com/
[9] https://www.lavamusic.com/

The development of SMIs has been aided by the availability of new embedded platforms: in a recent study, Meneses et al. [21] compared the different characteristics of three open-source embedded solutions for the implementation of an augmented musical instrument. These solutions were the Bela framework [22], Prynth [23], and a custom sound processing unit. The authors presented a clear overview of the advantages and drawbacks of each platform, resulting in each being found valid and apt for different applications. Along the same lines, Vignati et al. [24] compared different real-time architectures based on the Linux operating system, and found that the best performances were achieved by the use of the Xenomai kernel.

The Xenomai-based real-time operating system Elk Audio OS [25] was designed to streamline the development of digital hardware music devices such as SMIs, and has proven to be valid for several of these projects (e.g., the Sensus Smart Guitar, the SOURCE sampler [4], GuitarML's Neural Pi[10]). Our previous work reported in [26] successfully employed Elk Audio OS in the deployment of audio processing software involving deep learning inference. Having found Elk Audio OS to provide several tools that eased the development of real-time audio processing and recognition software for embedded devices, it was selected for the current study.

2.3 Guitar technique recognition

Playing technique recognition is a research topic that has received considerable interest in MIR throughout the years [27–29]. In the case of guitar, technique recognition has been investigated in the offline context for tasks such as extended music annotation [30], [31] and arpeggio identification [32]. In this case, technique recognition is performed on whole performance recordings.

Research on guitar playing technique detection has evolved from basic analyses to more advanced machine learning approaches. Early work by Traube et al. [33] and Penttinen et al. [34] established fundamental methods for plucking point detection, which significantly affects the timbral characteristics of guitar sounds. Subsequently, researchers explored various algorithmic approaches with mixed success: Chen et al. [35] achieved only moderate accuracy (74% *F*-score) with their two-stage algorithm for detecting five guitar techniques in solos, while Abeßer et al. [36] demonstrated that Gaussian Mixture Models outperformed other conventional machine learning classifiers for bass guitar playing style detection. More recent work has focused on sparse coding methods, with Su et al. initially achieving modest results (71.7% F1-score across seven techniques) [37], before improving performance to 79.72% *F*-score [38]. However, these approaches have generally been limited to offline analysis of pre-recorded monophonic guitar segments.

Offline algorithms are often able to achieve high result quality due to the availability of the entire context, i.e. audio signal, before, during, and after each note in the signal. However, offline recognition cannot be employed in real-time performance scenarios as it would require the performance to be over (or parts of it) to produce its results [39].

Differently from these, Reboursière et al. [40] represents one of the first approaches introducing the concept of real-time guitar-technique classification for controlling or triggering any external type of media or effects. The authors presented preliminary results on the detection of seven guitar techniques. Separate signal processing and feature extraction operations were used for each technique. The resulting set of custom detectors showed a detection success rate of about 90% across the different techniques. They also proposed an embedded hardware implementation, but not all the detection algorithms were adapted for real-time usage. Some of the authors improved on the performance of the system in [41], but the algorithms for each technique were not combined successfully and the results were obtained with the offline implementations. The different algorithms were reportedly grouped in a plugin with a later effort [42].

Some of the aforementioned studies approached guitar technique classification using conventional machine-learning solutions in real-time contexts. In contrast, others employed deep learning solutions for offline classification, and a few proposed embedded computing solutions. However, we found a lack of solutions that effectively combine these ideas for real-time technique recognition on resource-constrained devices such as embedded computers. Moreover, most of the works do not mention performer-instrument group isolation between training and testing, which has deep implications on performance metrics and generalization capabilities of a recognition system, and is often adopted in other works outside of guitar playing technique recognition [43–45].

2.4 Percussive guitar technique recognition

Approaches to real-time percussive hit classification and sound triggering have been approached in many studies, which focused on different sources as "interfaces", such as drums (e.g., the Cajón [46]) or everyday objects and surfaces [47, 48]. A similar approach was applied to the guitar by Lähdeoja [49], fitting the instrument with many piezoelectric transducers and performing onset and timbre detection on a laptop.

[10] https://github.com/GuitarML/NeuralPi

More recently, Martelloni et al. [50] conducted an interview study focused on the popular percussive fingerstyle for acoustic guitars. The style involves the use of the body of the guitar itself as a percussive instrument to accompany a solo performance. The study observed percussive fingerstyle guitarists in different conditions and investigated possible guitar augmentations that could benefit their performance. This resulted in a map of the locations of the most common percussive interactions. Moreover, the authors identified common patterns in the needs of the players which suggested that percussive fingerstyle performances can benefit from real-time classification of the percussion area utilized, which can be used to trigger substitute sounds or control the processing of the guitar signal.

The authors went ahead to implement an augmented guitar prototype [51] that was paired with three piezoelectric transducers. The signal from the transducers was fed to a timbre classification system with two output classes: one for kick-drum-like gestures and one for all other percussive gestures. The classification output triggered different sound samples depending on the predicted class. The authors further investigated the detection of percussive techniques on the guitar in [52], fitting the instrument with two more piezoelectric transducers (five in total, located at the main hit areas) and exploring the hierarchical classification of both the hit area and hand part used.

The system designed by Martelloni et al. involves the modification of an acoustic guitar with five transducers, which can greatly simplify the problem of percussive-hit classification by allowing classifiers to rely on time and loudness cues at different transducers to predict hit areas. This relies on multiple analog-to-digital converters (ADCs) and a laptop computer that runs a multichannel hit classifier in real-time. However, we seek a solution that is more cost-effective and minimally intrusive.

In the present study, we approach both percussive and pitched technique detection from the sole audio signal coming from the transducers fitted into amplified steel-string acoustic guitars. Moreover, we choose not to rely on a laptop or desktop computer for two main reasons: (i) first and foremost, because the recognition pipeline is meant to be part of an IoMusT device such as a smart guitar, and second (ii) because a solution solely based on small and embeddable SBCs can easily be part of either the instrument or an external device (e.g., a guitar effect pedal), making for an overall less cumbersome setup (i.e., one vs multiple transducers, one vs multiple shielded audio cables, one vs multiple preamplifiers and ADCs).

3 Methodology

This section describes the proposed playing technique approach and implementation, and is organized as follows: Section 3.1 provides information about the hardware. Section 3.2 describes the software pipeline, including onset detection, feature extraction, and technique classification. Finally, in Section 3.3, we present relevant considerations on latency and potential tradeoffs with accuracy.

3.1 Hardware

The classifiers trained for these studies were deployed on a Raspberry PI4 4 Gb SBC running the Elk Audio OS [25]. The choice of this SBC is justified by the compact size of the hardware and reasonable power consumption, along with the rather satisfactory computational power of this most recent model. For practical prototyping reasons, the Raspberry Pi was not inserted into the instrument but kept attached to the shoulder strap. Power was provided to the board via a USB power bank. The input and output audio signals were handled with the Elk-PI Hat development board for AD/DA conversion. The same results were obtained with a more compact HifiBerry DAC+ ADC Pro hat (see Fig. 1).

3.2 System architecture

In the proposed system, expressive guitar technique recognition is performed on a per-note basis, and the classification is composed of three separate steps, i.e., *onset detection*, *feature extraction*, and *classification* (See Fig. 3).

Onset detection is tuned on guitar sounds and used to trigger feature extraction, which is delayed to gather a sufficient amount of audio signal. Recent approaches in offline MIR employ end-to-end networks that operate

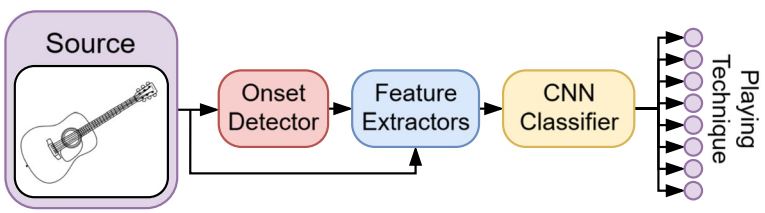

Fig. 3 Playing technique recognition pipeline. Input audio is fed to both an onset detector and several feature extractors. Upon detection of new onsets, the former triggers the latter, which computes features on a short audio context and feeds them to the deep classifier

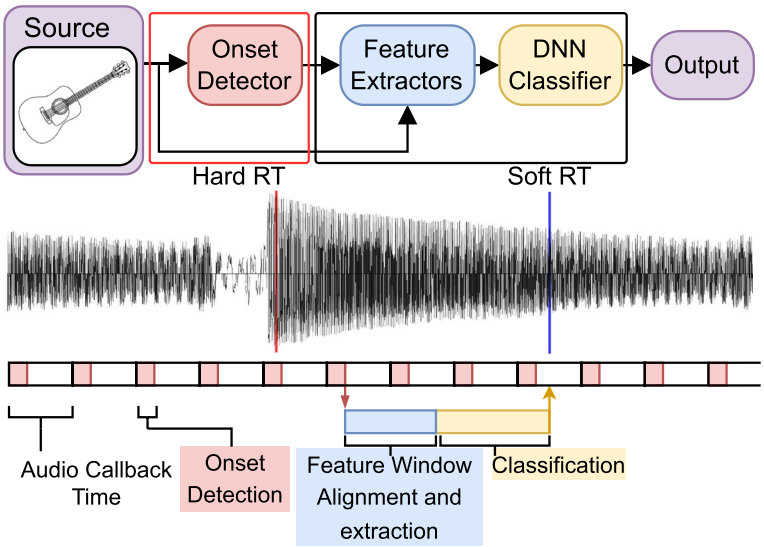

Fig. 4 Recognition pipeline along with an example of execution. Upon onset detection, after a small delay from the real onset, feature computation and classification are executed on a separate thread and can take longer than a single audio processing callback. This is allowed as note events are less frequent than the set audio callback rate for the project (64 samples at 48 kHz, resulting in 1.66 ms between audio callbacks, while note onset intervals are limited to a minimum of 20 ms through the detector's parameters)

directly on the raw audio signal and learn latent features. However, these are often rather computationally expensive and can be inapplicable in real-time embedded approaches that try to limit computation requirements and latency [53]. On the contrary, timbral features such as Mel Frequency Cepstral Coefficient (MFCC) can be extracted in real-time with a small delay and amount of computational power. Extracted features are then fed to a Convolutional Neural Network (CNN) to predict a higher-level property such as guitar technique starting from timbral features. Four CNNs were trained for Experiment 1, as will later be explained in detail.

An added advantage of a multistep pipeline is that it can be split into the steps that are required to run at the audio callback rate in the real-time thread (e.g., onset detection and audio buffering) and those that can run on a separate thread, only when an onset is detected (e.g., feature extraction and classification), see Fig. 4.

3.2.1 Onset detection

For onset detection, we used the Aubio library [54] implementation of the Modified Kullback–Leibler (MKL) distance function without the default adaptive whitening [39], whose parameters were optimized for both accuracy and latency with an evolutionary algorithm in [55]. The parameter tuning in [55] was performed with an earlier version of the playing guitar technique dataset used here (i.e., the AG-PT-set). On the current version of the main dataset, the optimized detector obtained an accuracy of 90.98% and an F1-score of 95.28%. On the smaller extra test dataset, the accuracy and F1-score of the onset detector were 95.72% and 97.81%, respectively. The playing technique recognition pipeline here was run with an audio buffer size of 64 samples and a sample rate of 48 kHz for compatibility with the optimized parameter assignation found in [55].

3.2.2 Feature extraction

For the feature extraction step, we used a set of timbral extractors from the TimbreID Puredata library [56], which were converted to C++ classes and made compatible with the Juce framework for audio plugins. The feature extractors are available as open-source[11]. We selected many feature extractors that describe timbre (in terms of spectral and temporal content), as it is the main sound property affected by the use of the selected playing techniques across different pitches [41]. The extractors of choice were MFCC, Bark Frequency Cepstral Coefficient (BFCC), Bark Spectrum, Bark Spectral Brightness, Real Cepstrum, Peak Sample, Attack Time, and Zero Crossing Rate. Full feature vectors were then subject to feature selection with ANOVA [57], with the number of selected features being a parameter optimized through grid search. For each note, features are extracted throughout a *feature window* interval. The feature window is composed of several overlapping sub-windows, each of which is 256 samples long and overlaps by 50% with the neighboring

[11] https://github.com/CIMIL/cpp-timbreID

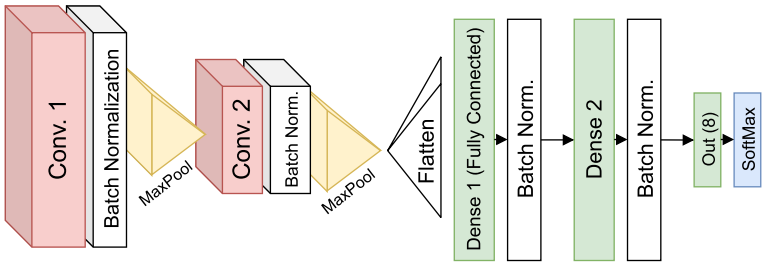

Fig. 5 General structure of the classification networks. First, a set number of 2D convolutional layers process the input feature matrix, and they are interleaved with batch normalization and pooling layers. Then, the data is flattened and passed through a set number of fully connected layers, which are also interleaved with batch normalization, and dropout is used for further regularization. Finally, the last layer applies the softmax function to the output. The numbers and types of hidden layers are part of the hyperparameters (see Table 2)

sub-windows[12]. The number of sub-windows ($\#SubW$) is determined by the overall feature window length following Eq. (1).

$$\begin{aligned} \#SubW &= \left\lceil \frac{n}{m \times or} + or \right\rceil \\ &= \left\lceil \frac{n}{256 \times 0.5} + 0.5 \right\rceil \; for\; n = 704, 2112, 3456, 4800 \end{aligned} \tag{1}$$

Where n is the length of the feature window in samples, m is the length of the overlapping sub-windows (i.e., 256), and or is the overlap rate (i.e., 50%).

For each overlapping sub-window, the aforementioned extractors produce a vector of 271 feature values[13]. Feature vectors are collected for each subwindow and arranged in a matrix of size ($271 \times \#SubW$).

The C++ feature extractors are included in both a recognition plugin that runs on the embedded computer, and a feature extraction plugin. The latter was run in offline mode in a Digital Audio Workstation (DAW) for faster-than-real-time extraction from the dataset.

3.2.3 Classification

The last step of the recognition pipeline consists of a CNN classifier, which is fed two-dimensional feature matrices from the initial part of each note and is trained to predict the playing technique used.

The general structure of the classifier includes a maximum of two convolutional layers, interleaved by batch normalization and pooling layers, followed by a maximum of two fully connected hidden layers, a fully connected output layer, and a final softmax activation. The following hyperparameters were then tuned through grid search: number of convolutional layers, number of fully-connected layers, number of input features for automatic selection, learning rate, batch size, training epochs, kernel size per layer, stride per layer, number of filters per layer, layer activations, pooling layer type, number of neurons per dense (i.e., fully connected) layer, and dropout rate. Multiple rounds of grid search were performed for each experiment condition, starting from a coarse grid and progressively refining the search space. The code and data used to train and test the classifiers are available on GitHub as well[14]. The base structure of the CNN classifier is represented in Fig. 5, and the parameter ranges for the grid search can be found in Table 1, along with the parameter values found for the four configurations of the first experiment.

The neural models used for each experiment were trained and tested using Keras and TensorFlow.

3.2.4 Embedded deployment

Each CNN was converted to the TensorFlow Lite format and deployed on the embedded board. The entire recognition pipeline was developed in C++ and compiled as a Virtual Studio Technology (VST) plugin to be run within Elk Audio OS through its headless DAW *Sushi*[15] [26]. The DAW was set to process audio at 48 kHz and with a buffer size of 64 samples, as it was found to grant a sufficiently low latency for onset detection and audio processing to be performed alongside the recognition pipeline [55]. Onset detection and feature extraction are then performed in the audio thread, while the more demanding classification through deep inference is performed on a separate thread so as not to interfere with the audio callback. Inference is performed using the TensorFlow Lite C++ library. Classification results are then reported in a non-locking fashion to the real-time audio thread, where they can be used for real-time synthesis

[12] The first sub-window is overlapped by 50% with the signal that comes before the beginning of the feature window, while the last overlaps with zero padding.

[13] The feature vector comprises the following coefficients and individual features: MFCC (38 coeff.), BFCC (50 coeff.), Bark Spectrum (50 coeff.), Bark Spectral Brightness (1), Cepstrum (129 coeff.), Peak sample (1), Attack Time (1), Zero Crossing Rate (1). Total = 271 values

[14] https://github.com/CIMIL/ExpressiveGuitar-TechniqueClassifier

[15] https://github.com/elk-audio/sushi

Table 1 Grid search parameter ranges

Hyperparameter	Start	End
Learning-rate	5×10^{-5}	1×10^{-3}
Batch-size	1	512
Training epochs	50	1500
Features (per subwindow)	50	271
Num. of conv layers	1	2
Conv. kernel sizes	3	5
Conv. strides	1	2
Conv. num. filters	8	32
Pooling layer types	[MaxPool,AvgPool]	
Num. of dense layers	0	4
Width of dense layers	16	100
Dropout rate	0.2	0.8

or sample triggering that is influenced by the prediction. The code for the recognition VST is made available on GitHub[16].

As a proof of concept, the classification result was used to synthesize a simple sine tone with a different pitch for each technique. Additionally, the results were sent via OSC messages over Wi-Fi to a computer with a simple visual graphics program developed in Processing, so that the use of different techniques changed the colors of projected visuals. Alternatively, a computer was set up with a DMX USB interface, OSC-enabled software, and connected to two moving-head lights and a smoke machine to trigger scenes or change to different rotation angles and colors depending on the playing technique recognized (see Fig. 2). This prototype of IoMusT environment serves as a proof-of-concept and will be further developed and analyzed in future works.

3.3 Latency

The target "repurposing" application of a real-time playing technique recognition system, i.e., the way that the classification results are used, determines the maximum tolerable latency for the user. In turn, this affects how long the system can analyze the audio signal from the onset of a note before having to produce a classification on the currently available context. In a sound-to-sound application, i.e., where the classification made on the incoming audio controls sound generation or processing, latencies as little as 10 ms can be the maximum tolerable delay between action and sound for players [58]. However, there are application scenarios where recognition results are used for less time-critical control parameters of a sound synthesis engine, while note triggering is performed through quicker pitch trackers and envelope followers [14]. In this case, the latency of the recognition system can be increased to allow for more accurate classification results without affecting the perceptual quality of the instrument. Other applications can include sound-to-video systems, where the use of different playing techniques can trigger or affect live visuals in conjunction with continuous measures such as signal amplitude or pitch. In such applications, video artists might desire a system with a latency that matches the frame rate of the video output, which is typically 30 or 60 frames per second (i.e., 33 or 16 ms). However, this would be a requirement only when using fast-paced video animations or effects. For more relaxed musical styles, musicians and visual artists may steer towards slowly evolving visuals, in which case they may be willing to trade off some latency for better recognition accuracy. This can also apply to applications that control stage equipment over the network, such as lighting, smoke machines, or other stage effects [7]. These requirements indicate that the end-users or the developers of these repurposing applications could benefit from real-time recognition systems that offer a flexible choice of the tradeoff between how accurate and how reactive the system is. In light of this, Experiment 1 involves the analysis of different accuracy-latency tradeoffs with the proposed recognition system.

To measure the onset-to-results latency for the proposed system, test data recordings were played on a computer and fed to the board through the output of a USB audio interface (Focusrite Scarlett 2i2). At the same time, the stereo output from the board was recorded through two inputs of the audio interface. The left channel contained a passed-through version of the guitar signal fed to the classifier, while the right contained a reference "spike" produced by the classifier along with its prediction for each note. Note onsets in the output recordings were then labeled manually with the Audacity[17] software. The total software latency of the system was measured as the time interval between each labeled note onset in the left channel and the relative spike in the right channel. The measurement excluded any hardware or buffering latencies as they vary depending on the DAC and ADC used and can be easily found in the relative datasheets (i.e., the hardware used resulted in a latency of 3.9 ms at a sample rate of 48 kHz and 64 samples per audio buffer). Different hardware of choice yields different input/output latency that can be summed to the software latency to obtain an end-to-end measure.

Latency was further broken down into its main components, namely the onset detection latency (T_{OD}), the feature-window alignment delay (T_{FWA}) introduced after onset detection to properly delay feature computation,

[16] https://github.com/CIMIL/cpp-timbreID

[17] https://www.audacityteam.org/

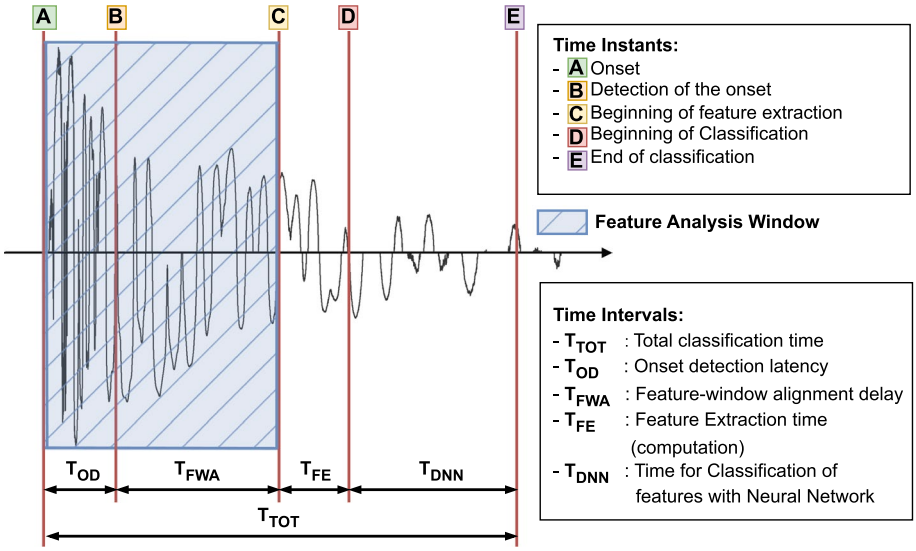

Fig. 6 Representation of the time intervals that compose the total latency of the system. The total onset-to-results latency corresponds to the sum of the onset detection latency, the feature-window alignment delay, the feature computation latency, and finally the classifier inference time

the feature extraction or computation latency (T_{FE}) and the classification latency (T_{DNN}). A graphical representation of the time instants and corresponding intervals is shown in Fig. 6.

4 Evaluation

4.1 Data

The dataset used for the experiments is the Acoustic Guitar Playing Technique dataset (AG-PT-set) that we presented in [59]. AG-PT-set is composed of over 10 hours of recordings with 32,592 individual notes captured with the internal pickups[18] of 6 electro-acoustic guitars. These notes are played with 8 distinct playing techniques (including percussive techniques) and three different dynamics (i.e., piano, mezzo forte, forte) by 7 experienced guitarists. The techniques considered for this study are the following:

1. "Kick" technique: hit on the lower right part of the guitar top;
2. "Snare-1" technique: hit on the lower side of the guitar body;
3. "Tom" technique: hit on the upper guitar body near the top of the fretboard end;
4. "Snare-2" technique: hit on the muted strings over the fretboard;
5. "Natural Harmonics": plucking the strings while lightly touching the string with the fretting finger, therefore forcing a node in the displacement of the string and letting only some harmonic overtones ring;
6. "Palm Mute": partially muting a string with the palm of the picking hand;
7. "Pick Near Bridge": plucking a string near the saddle (i.e., bridge);
8. "Pick Over the Soundhole": plucking a string over the soundhole.

Except for "Pick Near Bridge", all pitched sounds are produced by plucking strings over the soundhole. A more detailed description of the techniques and how the dataset was recorded and labeled can be found in [59].

Additionally, for *Experiment 3* (see Section 4.4), we recorded a small additional test dataset. This *extra-test* dataset is composed of 362 notes recorded by the pair Player-A/Guitar#1 (which is already present in the main dataset), and 380 notes recorded by the new pair Player-B/Guitar#1, where guitar Guitar#1 is the same as in the main dataset, but Player-B is a different guitarist playing the same guitar. Both guitarists were provided with the same expressive instructions. The extra-test dataset is used to measure differences in the recognition accuracy between the two players on the same guitar, which could suggest a non-negligible effect performer individuality on the recognition system. Performer individuality has been defined as idiosyncratic patterns of performance actions that can result in different timbral nuances among different performers [12]. In the case of guitar, performance actions that can result in different timbral nuances can be, for example, the amount of force applied in string muting, hand positioning, or pick positioning.

[18] Piezoelectric plus internal condenser microphone for 5 out of 6 guitars, and solely piezoelectric for the remaining.

4.2 Experiment 1: recognition accuracy under different latency constraints

Following the considerations on latency discussed in the previous section, we devised Experiment 1 to evaluate the accuracy achievable under different latency constraints. For this, we optimized and trained four classifiers with respectively four target latencies of 15, 45, 75, and 100 ms. In previous works, we observed how 15 ms is a reasonably small minimum latency that can provide adequate accuracy results with our approaches and data [59], while 100 ms was a reasonably large delay that we consider to be greater than the maximum tolerable latency for most musical tasks, even under the most relaxed constraints. The target latencies of 45 and 75 ms were chosen by equally splitting the range between the extremes and rounding. For each condition, the main effect of a different target latency is on the length of the feature extraction window. The longer the window, the more information is available to the classifier, but this also increases the latency. In this case, for the four conditions, the window length was set to 704, 2112, 3456, and 4800 samples respectively, with a sample rate of 48 kHz. Each value is an integer multiple of the audio block size used (i.e., 64 samples).

With the windowing procedure described in Section 3.2, the four experiment settings correspond to extracted feature matrices of sizes (271 × 6), (271 × 17), (271 × 28) and (271 × 38) for each note respectively.

For each condition, the classifier was optimized through grid search resulting in different CNN architectures. The best-performing model for each condition is shown in Table 2. Additionally, Fig. 5 shows the model structure. The accuracy was evaluated through a *Minus-1-PI* method (or leave-one-Performer/Instrument-pair-out, see Section 4.3), where the data was split into 7-folds corresponding to data from the 7 player/instrument pairs in the dataset. For each fold, the model was trained on the other 6-folds and tested on the selected data, so that the test player/instrument pair was never used for training. The resulting metrics are averaged over all folds.

Once the best model for each configuration was found, these were converted to the TensorFlow Lite format and deployed on the embedded board. The latency of the whole pipeline was measured by feeding guitar recordings to the input of the system while recording the output of the board. The output is composed of a copy of the clean input signal, and a reference signal, which allowed us to accurately measure the total latency. Furthermore, software probes that employed a steady clock allowed us to measure the separate latency components mentioned in Section 3.3

Table 2 Optimal values found through grid-search for the hyperparameters of each classifier configuration

	Feature window size			
Hyperparameter	**704**	**2112**	**3456**	**4800**
Learning-rate	8.0e−05	1.0e−04	1.0e−05	1.0e−04
Batch-size	64	64	128	128
Training Epochs	500	600	600	300
Features (per subwindow)	271	200[a]	200[a]	200[a]
Feature matrix shape[b]	[271×6]	[200×17]	[200×28]	[200×38]
Num. of conv layers	2	1	1	1
Conv. kernel sizes	3×3,3×3	5×5	5×5	5×5
Conv. strides	1,1	2	2	2
Conv. num. filters	4,4	32	64	32
Conv. activations	relu,relu	relu	relu	relu
Pooling layer types	Max,Max	Avg	Avg	Max
Num. of dense layers	2	0	0	0
Width of dense layers	32	16	16	16
Dropout rate	0.5	0.5	0.5	0.5
Model weights	10,404	52,168	181,128	116,168

[a] For the configurations 2112, 3456, and 4800, the best grid search results were obtained with feature selection with ANOVA [57] and 200 features per subwindow

[b] The number of subwindows follows Eq. (1)

4.3 Experiment 2: generalization performance and the performer-instrument effect

An ideal instrument-playing technique classifier with good generalizability should offer high recognition accuracy independently of different instrument characteristics, recording conditions, and different player's playing styles [43]. In data-based approaches to a rather complex task such as real-time technique recognition from few-millisecond-long analysis windows, generalizability can come down to the amount and diversity of the available data, as well as the fitting behavior of the approach.

Experiment 2 focuses on the generalization performance of the system. For this, we started from one of the configurations of Experiment 1 (i.e., the lowest latency, 704 samples configuration) and trained and tested the classifier starting from a single performer/instrument pair and progressively adding the remaining pairs one at a time. This was done to assess whether the amount of data available to us is a limiting factor for the performance of the system and to roughly estimate the amount of data required to achieve a certain level of performance.

Moreover, since preliminary results with stratified 5-fold cross-validation displayed potentially over-optimistic results, we set to verify whether the current task is subject to confounding factors [9] akin to the artist and album effects [10, 11]. Album and artist effects refer to the artificial inflation of classification performances

that can be caused by the presence of the same artists or albums in both training and testing recordings. In particular, approaches that fit to irrelevant artist or album characteristics (e.g., mastering chain or a specific studio reverb) can obtain extremely over-optimistic results with improper evaluation, while failing to generalize. In the case of the current task, the performer-instrument pairs in the dataset can be identified as natural data groups (similar to the artist for genre classification) and can potentially represent a confounding factor if not properly separated. We refer to the potential effect as performer-instrument effect and set out to assess whether it has an impact for the current task.

To do this, we separated natural groups in the data, avoiding performer-instrument overlap between the training and test set. For this, we used a *Minus-1-PI* method, adapted from Livshin's Minus-1-DB method [43]. With *Minus-1-PI* method (Minus one Player/Instrument pair), the data is split into 7-folds corresponding to the 7 player/guitar pairs in the dataset. In this sense, *Minus-1-PI* evaluation is a special case of Group K-Fold cross-validation, where each fold contains exactly one performer/instrument pair and is therefore akin to a leave-one-out strategy. This way, the model is not only tested on data that was not seen during the training, but rather data that was played by a different player/guitar pair than those in the training set. *Minus-1-PI* results were compared to the more common stratified 5-fold cross-validation measurement. For this, the sci-kit learn `StratifiedKFold` utility was used to create 5-folds where the percentage of samples from each class was preserved from the original dataset.

4.4 Experiment 3: specialization performance and performer individuality

Once we assessed the generalization capabilities of the system, we set to measure the performance of a specific single instrument. It is not unreasonable in fact, for an embedded recognition system for a SMI to target a specific instrument rather than focusing on performance across different instruments. In this context, we set to verify whether adding data from different performer/instrument pairs would affect recognition accuracy. This can be a useful result because, in the case that the data from a single instrument is limited, it can inform whether it is possible to integrate data from different conditions, or if it is necessary to collect more data. To do this, we recorded extra data (see Section 4.1) for Guitar#1 and a player that we will refer to as Player-A from now on. The pair Player-A/Guitar#1 was already present in the main dataset. The final model for the chosen configuration (i.e., 704 samples feature window) was first trained on the main dataset recordings for only the pair Player-A/Guitar#1, and tested on the extra data for the same pair. Then, the model was trained and tested six more times by progressively adding the six remaining pairs to the training set only.

Additionally, we set to evaluate the performance of these trained models on the same instrument (i.e., Guitar#1) but with a different player (i.e., Player-B) that was not present in any recording of the main dataset. Although limited in its extent, this experiment can help to understand whether the performance of a real-time playing guitar technique recognition system can be affected by a certain degree of performer individuality. Performer individuality describes subtle differences in the way different performers play the instrument, resulting in idiosyncratic timbral nuances [12]. Similar performer-based features have been studied in the past and even used for player identification [60] While the existence of a degree of performer individuality and its repercussions on the instrument's sound are undoubted, it is yet to be verified whether the differences in the way a musician intends and plays certain techniques can affect the very first milliseconds of played notes. Moreover, it is yet to be verified whether the extent of these differences can affect the performance of a timbre-based recognition system. This can help inform a future in-depth study on the matter that could, in turn, drive relevant choices in how to integrate new recordings into our dataset, or the creation of new datasets that are specifically targeted to a single instrument.

5 Results and discussion

5.1 Experiment 1: latency and accuracy

The results of Experiment 1 are shown in Figs. 7 and 8. The first plot shows the recognition accuracy across the eight techniques and the latency of the different configurations of the system, while the second shows the F1-score for each technique. Accuracy and F1-score values reported are averaged over 7-fold with the *Minus-1-PI* approach (see Section 4.3). The total latency is broken down into its components, as described in Fig. 6.

The accuracy results show a clear increasing trend with the increase of the feature window (therefore latency). However, the greatest increase in accuracy is observed between the 704 and 2112 window configurations, with marginal improvements with longer windows. Figure 8 offers better insight into why this is the case. In particular, we see how the performance for all techniques increases between the first two configurations, indicating that 704 samples at 48 kHz may be a very limited context to capture most of the playing techniques, but the most sizable performance jumps are seen with pitched techniques and natural harmonics in particular. This can be attributed to the more complex timbral nature of pitched techniques,

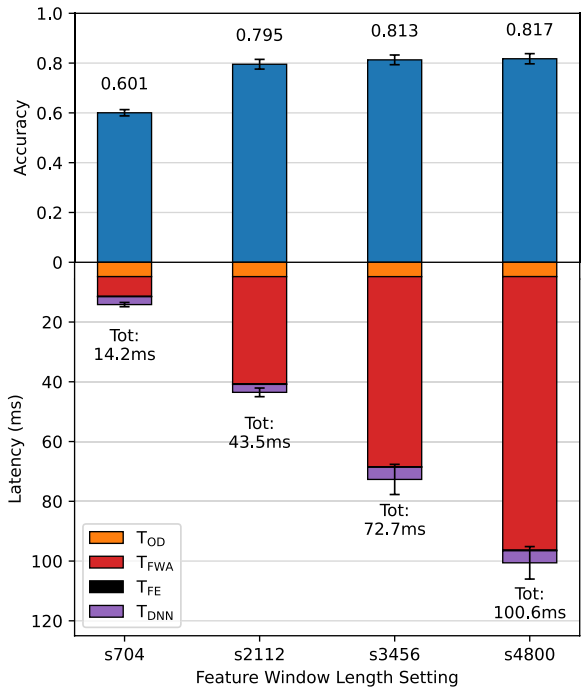

Fig. 7 Recognition accuracy and latency of the four configurations for Experiment 1. For each configuration, a different feature extraction window length is used, impacting both the accuracy and latency of the system. The total latency is reported along with a breakdown of its different components, as described in Fig. 6. Error bars for the accuracy represent the standard error across evaluation folds, while for the total latency, they represent the standard error across repeated measurements. Detailed measures of the latency components can be found in Table 3

with common spectral cues being spread across a wide range of pitches in the data.

Moreover, natural harmonics are played with a fast action that can be, at times, detected as two onsets, making it difficult to align the feature window. While the first action is always detected as the main onset, and the second suppressed by a debouncing mechanism, the more technique-related timbre content happens mainly after the second onset. This can explain how longer windows, which include this additional context, improve the performance with natural harmonics. A more in-depth analysis of the natural harmonic false negative notes shows that these are generally confused with other pitched techniques, and to a lesser degree with the Snare-1 technique. An example of the occasional double onset is displayed in Fig. 9.

With longer feature windows, the performance of pitched techniques tends to increase less and stagnates in some cases, but the performance of percussive techniques tends to decrease. This can be because percussive techniques generate sounds with shorter decay and more information around the early attack phase of the signal. While adding more context should not affect the performance of percussive techniques, trying to also classify pitched sounds may cause the network to favor focusing on the whole feature window instead of the sole attack.

In terms of the latency results measured on the embedded implementation, the impact of the classifier inference (T_{DNN}, Fig. 6) becomes less relevant with greater feature windows. Moreover, most of the latency in the configurations with 2112, 3456, and 4800 is constituted by the feature-window alignment delay (T_{FWA}), which is used to align the feature windows with the onset and depends

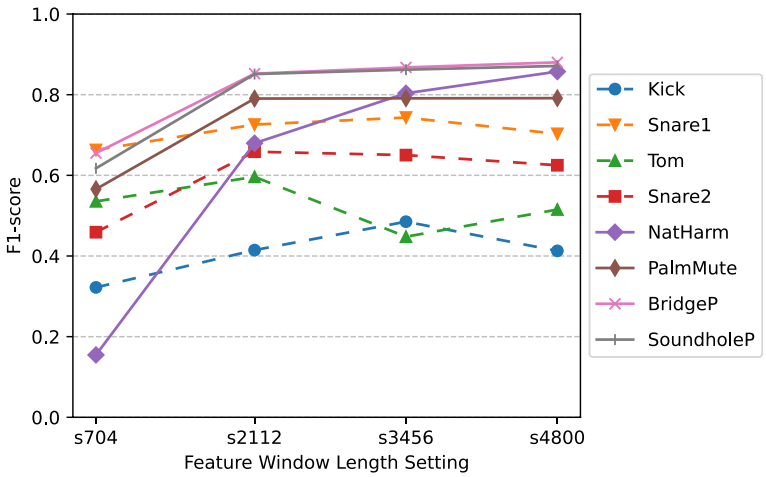

Fig. 8 F1-score for each playing technique and each configuration of Experiment 1. Dotted lines represent percussive techniques, while solid lines represent pitched techniques (see Section 4.1)

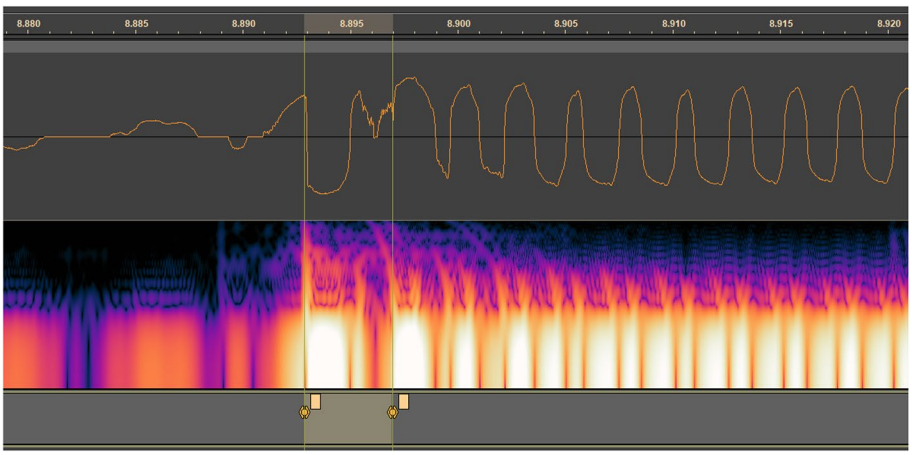

Fig. 9 Example of a natural harmonic sound where two onsets that are 4 ms apart can be distinguished. The first can be attributed to the picking gesture, while the second is followed by a harmonic behavior

on both its size and T_{OD}. The onset detection delay (T_{OD}) remains the same as the configuration of the detector was not changed, but the composition of the total latency shows how we could have different onset detector settings with more latency and greater accuracy and reduce T_{FWA} without affecting the total latency. Finally, the time required by the actual computation of the feature matrices (T_{FE}) is negligible, as most of the computations are simple and performed when each audio block of 64 samples is added to the buffer (Table 3).

5.2 Experiment 2: generalization performance and the performer-instrument effect

The results of Experiment 2 are shown in Figs. 10 and 11. The plots show the recognition accuracy of the 704 sample feature window configuration, trained and tested on a progressively larger dataset, where performer/instrument pairs are added one at a time. Figure 11 represent the average results obtained with regular stratified 5-fold cross-validation, where all the data from the selected performer/instrument pairs is mixed and each sample is eligible to become part of any of the 5-folds. Conversely, Fig. 10 represents the results of *Minus-1-PI* evaluation, where data from each performer/instrument pair constitute a separate fold. As a consequence, the rightmost bar of Fig. 10 corresponds to the first bar of Fig. 7, while the remaining bars represent the performance of the model trained on fewer data. The takeaways from the results of Experiment 2 are the following:

1. Generalization performance increases with the addition of data from more guitars and players, but the current size of the dataset is limiting;
2. The performance averaged over stratified 5-fold cross-validation is over-optimistic and misleading as it decreases with the increase of the real generalization capabilities of the classifier.

Takeaway 1 highlights an expected increase in the performance of the model, but it indicates that the task of generalization over multiple guitars and players is complex, and having only seven performer/instrument pairs in the dataset is limiting the potential recognition

Table 3 Detailed latency breakdown of all the recognition configurations with relative standard deviations

Model	T_{OD} (ms)	T_{FWA} (ms)	T_{FE} (ms)	T_{DNN} (ms)	T_{TOT} (ms)
s704	4.8 ± 0.7	6.6 ± 0.0	0.1 ± 0.0	2.6 ± 0.0	14.2 ± 0.7
s2112	4.8 ± 0.7	35.9 ± 0.7	0.2 ± 0.0	2.7 ± 0.0	43.5 ± 1.5
s3456	4.8 ± 0.7	63.6 ± 4.2	0.2 ± 0.0	4.1 ± 0.2	72.7 ± 5.0
s4800	4.8 ± 0.7	91.5 ± 4.6	0.2 ± 0.0	4.0 ± 0.1	100.6 ± 5.4

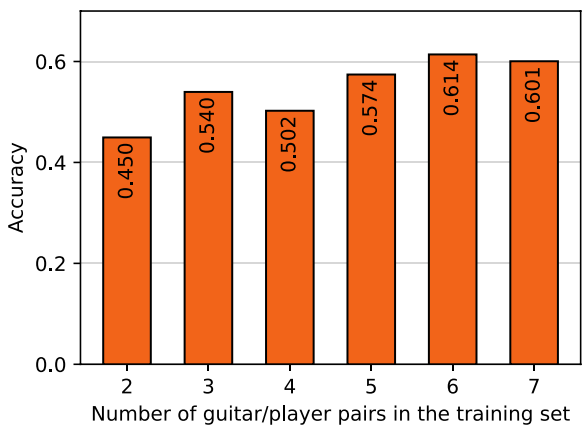

Fig. 10 Generalization results comparing accuracy with the increase of the number of guitars in the dataset, using the *Minus-1-PI* approach for cross-validation

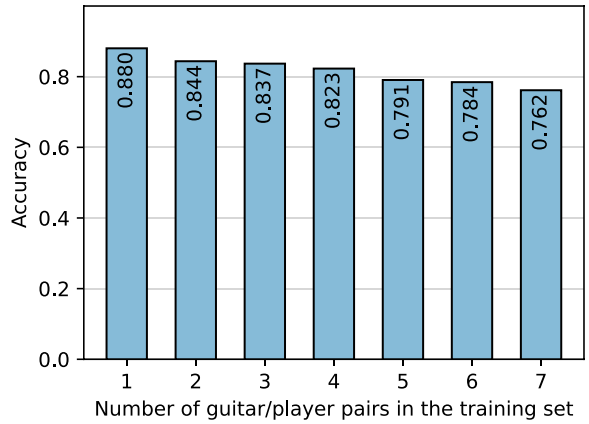

Fig. 11 Generalization results measured with a common but incorrect validation procedure (Stratified 5-fold cross-validation)

performance. Furthermore, the unstable increase in performance might suggest that, for this task, a satisfactory number of performer/instrument pairs would be considerably higher than seven. These results suggest how future efforts should either be directed at increasing the size of the dataset, or at investigating the possibilities of specializing the trained models on a single instrument with techniques such as layer-freezing and fine-tuning, or domain adaptation. Using a small and possibly unlabeled set recorded on the fly would be a feasible and practical way to adapt the pre-trained model in a real-world scenario.

Takeaway 2 highlights how using a cross-validation procedure that is not grouped by performer/instrument pair can be over-optimistic and misleading for the current task, as it shows a decreasing trend in accuracy with the increase of actual generalization capabilities of the model. Moreover, the accuracy from an incorrectly split evaluation is the highest with a single guitar and player pair between the train and test sets, where the test can benefit the most from unwanted overfitting to the specific characteristics of the single pair in the dataset.

This becomes especially detrimental to the real generalization performance of the classifier when recognition accuracy measured with non-grouped cross-validation is used to drive hyperparameter optimization and model selection. In this case, we observed the insurgence of a performer-instrument effect that fosters the overfitting of the model to the specific characteristics of the known guitars and players rather than learning the general properties of the playing techniques.

5.3 Experiment 3: specialization performance and performer individuality

The results of Experiment 3 are illustrated in Fig. 12. The plot shows the recognition accuracy of the 704 sample feature window configuration, trained starting from a specific performer/instrument pair (i.e., Player-A/Guitar#1) and tested on extra data from the same instrument but two distinct players (i.e., Player-A and Player-B). Furthermore, the classifier is progressively trained on data from the other performer/instrument pairs in the main

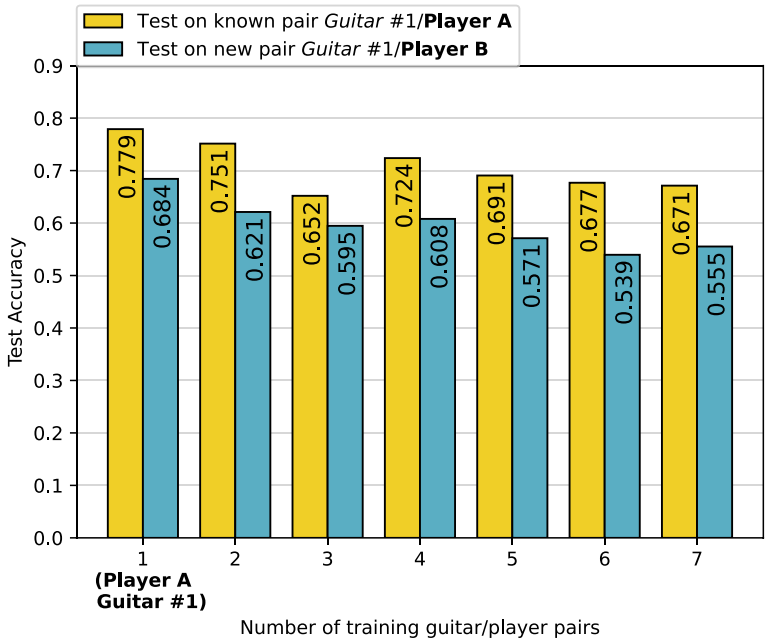

Fig. 12 Test accuracy with one guitar (1), a known player (A), and a new player (B)

dataset. While Player-A is always in the training set for each model, Player-B did not record any data for the main dataset. The main results of Experiment 3 are the following:

A The model trained only on data from Player-A/Guitar#1 performs considerably better in extra data from the same pair than on new performer/instrument pairs (see Experiment 1 results in Section 5.1);
B Adding data from other performer/instrument pairs to the training set is detrimental to the performance of the model on the specific Player-A/Guitar#1 pair. This indicates that specific data is preferable over data quantity;
C Most importantly, the model trained on Player-A/Guitar#1 performs consistently worse when a new (unknown) Player-B plays the same instrument. This suggests that performer individuality can be a relevant factor for the performance of a playing guitar technique classifier.

Takeaway A confirms that the classifier retains good generalizability from recording factors, as the extra test set was recorded months later, with a different pick, cables, and slightly different guitar settings (e.g., neck bow, volume setting). Additionally, this can highlight how the generalization metrics of Experiment 1 are not relevant when designing a classifier for a specific instrument and player. Takeaway B instead highlights how adding training data from new performer/instrument pairs cannot help the specialization performance for a target pair. Takeaway C suggests that the ways that two musicians play the same techniques differently (i.e., performer individuality) can significantly affect the recognition performance of a real-time playing guitar technique classifier, even with the very short context considered. While the limited extra data used for Experiment 3 does not allow for a deeper analysis of performer individuality on the task, the consistency of the results suggests that this factor is relevant. While it is easy to find reasons why different guitars can trick the classifier, further study will need to be devoted to understanding why performer individuality exists for guitar and has such an impact.

6 Conclusions

In this paper, we presented a flexible-latency embedded real-time playing guitar technique recognition pipeline for a smart acoustic guitar. The system was designed for low-latency recognition resource-constrained embedded devices, and it was implemented and successfully deployed to a Raspberry Pi 4 with the Elk Audio OS. As a proof-of-concept, we integrated the smart guitar into a networked environment, where classification results were used to control live visuals and stage equipment.

Moreover, we investigated the impact of the task requirements and data characteristics on recognition performance. We found that relaxing latency constraints, especially between 15 and 45 ms, can benefit the recognition accuracy for pitched and percussive techniques, while the performance for percussive techniques is mostly unaffected and even slightly degraded with larger feature windows. Additionally, we observed a tendency for models to fit specific guitar and player characteristics, limiting their ability to learn the broader properties of playing techniques and the generalization performance on new instruments and players. We successfully addressed the issue by employing a *Minus-1-PI* evaluation method to ensure that guitar and player pairs remain separate during training and testing, which provides a reliable accuracy metric to drive hyperparameter optimization. On the contrary, the accuracy metrics obtained with simple k-fold cross-validation were revealed to be misleading as they decreased with the increase in real generalization performance. Finally, we focused on a single instrument and found how this can expectedly lead to better performance, but also how performer individuality manifested in the different touch, or style, of different guitarists can affect the recognition performance.

We acknowledge important limitations in our study that provide clear directions for future research. First, the additional dataset used to investigate performer individuality was constrained in size, which potentially limits the comprehensiveness of our findings on individual playing styles. The research primarily focused on attack-based techniques in monophonic signals, representing a narrow subset of guitar-playing techniques and signal complexity. This excludes playing techniques such as vibrato or hammer-on. Polyphony will be addressed with the use of a hexaphonic pickup, and we will investigate embedded real-time recognition of a broader range of guitar techniques by integrating additional features across multiple time scales. Future research will also explore different classification structures, such as parallel models to separate percussive and pitched techniques and improve performances. Ultimately, we aim to conduct a more thorough investigation of the performer individuality through the integration of new data in our expressive guitar technique dataset, and in-depth analysis of signal differences.

Acknowledgements
Not applicable.

Authors' contributions
DS: conception, code implementation, network training, main writing, and research oversee. LT: conception, research oversee, manuscript contribution, and revision. All authors read and approved the final manuscript.

Funding
Not applicable.

Data availability
The datasets used to train the models in this paper are available at https://doi.org/10.5281/zenodo.10159492 and described in [59]. The code used to train the models is available at https://github.com/CIMIL/ExpressiveGuitar-TechniqueClassifier.

Declarations

Competing interests
LT is a co-founder of Elk Audio AB and a guest editor for the current special issue: Signal Processing for the Internet of Sounds.

Received: 9 December 2024 Accepted: 16 June 2025
Published online: 17 July 2025

References

1. L. Atzori, A. Iera, G. Morabito, The Internet of Things: A survey. Comput. Netw. **54**(15), 2787–2805 (2010). https://doi.org/10.1016/j.comnet.2010.05.010
2. G. Widmer, D. Rocchesso, V. Välimäki, C. Erkut, F. Gouyon, D. Pressnitzer, H. Penttinen, P. Polotti, G. Volpe, Sound and music computing: Research trends and some key issues. J. New Music. Res. **36**(3), 169–184 (2007). https://doi.org/10.1080/09298210701859222
3. L. Turchet, C. Fischione, G. Essl, D. Keller, M. Barthet, Internet of musical things: Vision and challenges. IEEE Access **6**, 61994–62017 (2018). https://doi.org/10.1109/ACCESS.2018.2872625
4. F. Font, in *Proc. 16th Int. Audio Mostly Conference*. SOURCE: a Freesound Community Music Sampler (2021), pp. 182–187. https://doi.org/10.1145/3478384.3478388
5. A. Migicovsky, J. Scheinerman, G. Essl, in *Proc. Int. Computer Music Conference (ICMC)*. MoveOSC - Smart Watches in Mobile Music Performance (Michigan Publishing, Ann Arbor, 2014), p. 692–696
6. S. Thorn, in *Proc. 16th Int. Audio Mostly Conference*. Telematic wearable music: Remote ensembles and inclusive embodied education (2021), pp. 188–195. https://doi.org/10.1145/3478384.3478386
7. L. Turchet, Smart musical instruments: Vision, design principles, and future directions. IEEE Access **7**, 8944–8963 (2019). https://doi.org/10.1109/ACCESS.2018.2876891
8. M. Romani, G.A. Giudici, D. Stefani, D. Zanoni, A. Boem, L. Turchet, in *Proc. 5th Int. Symposium on the Internet of Sounds (IS2)*. BCHJam: a Brain-Computer Music Interface for Live Music Performance in Shared Mixed Reality Environments (2024), pp. 1–9. https://doi.org/10.1109/IS262782.2024.10704087
9. F. Rodríguez-Algarra, B.L. Sturm, S. Dixon, Characterising confounding effects in music classification experiments through interventions. Trans. Int. Soc. Music. Inf. Retr. (2019). https://doi.org/10.5334/tismir.24
10. E. Pampalk, A. Flexer, G. Widmer, et al., in *Proc. 6th Int. Society for Music Information Retrieval Conference (ISMIR)*. Improvements of audio-based music similarity and genre classificaton, vol. 5. (ISMIR, London, 2005), pp. 634–637
11. A. Flexer, D. Schnitzer, Effects of album and artist filters in audio similarity computed for very large music databases. Comput. Music. J. **34**(3), 20–28 (2010)
12. M. Bernays, C. Traube, Investigating pianists' individuality in the performance of five timbral nuances through patterns of articulation, touch, dynamics, and pedaling. Front. Psychol. **5**, (2014). https://doi.org/10.3389/fpsyg.2014.00157
13. A.R. Jensenius, M.J. Lyons, *A NIME Reader: Fifteen Years of New Interfaces for Musical Expression. Current Research in Systematic Musicology*, vol. 3 (Springer, Cham, 2017). https://doi.org/10.1007/978-3-319-47214-0
14. M. Puckette, in *Pure Data Convention, Montreal*. Patch for guitar (PureData Convention Collective, 2007), p. 1–5
15. N. Bouillot, M. Wozniewski, Z. Settel, J.R. Cooperstock, in *Proc. Int. Conf. on New Interfaces for Musical Expression (NIME)*. A Mobile Wireless Augmented Guitar (Genoa, 2008), pp. 189–192. https://doi.org/10.5281/zenodo.1179499
16. L. Reboursière, C. Frisson, O. Lähdeoja, J.A. Mills, C. Picard-Limpens, T. Todoroff, in *Proc. Conf. on New Interfaces for Musical Expression (NIME)*. Multimodal Guitar: A Toolbox For Augmented Guitar Performances (Sydney, 2010), pp. 415–418. https://doi.org/10.5281/zenodo.1177881
17. I. Angulo, S. Giraldo, R. Ramirez, in *Proc. Int. Conf. on Technologies for Music Notation and Representation (TENOR)*. Hexaphonic guitar transcription and visualization (2016), pp. 187 – 192. https://doi.org/10.5281/zenodo.1289596
18. O. Lähdeoja, in *Proc. 12th Int. Conf. in Sound and Music Computing, SMC 2015*. An Augmented Guitar With Active Acoustics (2015), pp. 85–89. https://doi.org/10.5281/zenodo.851049
19. E.A. Meneses, S. Freire, M.M. Wanderley, in *Proc. Int. Conf. on New Interfaces for Musical Expression (NIME)*. GuitarAMI and GuiaRT: two independent yet complementary augmented nylon guitar projects (Virginia Tech, Blacksburg, 2018), p. 222–227
20. D. Stefani, Embedded Real-time Deep Learning for a Smart Guitar: A Case Study on Expressive Guitar Technique Recognition. Ph.D. thesis, University of Trento (2024). https://doi.org/10.15168/11572_399995
21. E. Meneses, J. Wang, S. Freire, M. Wanderley, in *Proc. Int. Conf. on New Interfaces for Musical Expression (NIME)*. A comparison of open-source linux frameworks for an augmented musical instrument implementation (2019), pp. 222–227. https://doi.org/10.5281/zenodo.3672934
22. A. McPherson, V. Zappi, in *Proc. Audio Engineering Society 138th Convention*. An environment for submillisecond-latency audio and sensor processing on beaglebone black (Audio Engineering Society, New York, 2015)
23. I. Franco, M.M. Wanderley, in *In Proc. 12th Int. Symposium on Computer Music Multidisciplinary Research (CMMR)*. Prynth: A framework for self-contained digital music instruments (Springer, Cham, 2016), p. 357–370
24. L. Vignati, S. Zambon, L. Turchet, A comparison of real-time Linux-based architectures for embedded musical applications. J. Audio Eng. Soc. **70**(1/2), 83–93 (2022)
25. L. Turchet, C. Fischione, Elk audio os: an open source operating system for the internet of musical things. ACM Trans. Internet Things **2**(2), 1–18 (2021)
26. D. Stefani, L. Turchet, in *4th International Symposium on the Internet of Sounds (IS2)*. Real-time embedded deep learning on elk audio os (2023), pp. 21–30. https://doi.org/10.1109/IEEECONF59510.2023.10335204
27. V. Lostanlen, J. Andén, M. Lagrange, in *Proc. 5th Int. Conf. on digital libraries for musicology*. Extended playing techniques: the next milestone in musical instrument recognition (Association for Computing Machinery, New York, 2018), p. 1–10
28. C. Wang, E. Benetos, V. Lostanlen, E. Chew, Adaptive scattering transforms for playing technique recognition. IEEE/ACM Trans. Audio Speech Lang. Process. **30**, 1407–1421 (2022). https://doi.org/10.1109/TASLP.2022.3156785
29. C. Wang, V. Lostanlen, E. Benetos, E. Chew, in *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*. Playing technique recognition by joint time–frequency scattering (IEEE, 2020), pp. 881–885. https://doi.org/10.1109/ICASSP40776.2020.9053474
30. T.H. Özaslan, E. Guaus, E. Palacios, J.L. Arcos, in *Proc. 7th International Symposium on Computer Music Modeling and Retrieval (CMMR)*. Attack based articulation analysis of nylon string guitar (Springer, Berlin, 2010)
31. C. Kehling, J. Abeßer, C. Dittmar, G. Schuller, in *Proc. 17th Int. Conf. on Digital Audio Effects (DAFx 2014)*. Automatic tablature transcription of electric guitar recordings by estimation of score- and instrument-related parameters (International Audio Laboratories Erlangen, Erlangen, 2014), pp. 1–8
32. I. Barbancho, G. Tzanetakis, A.M. Barbancho, L.J. Tardón, Discrimination Between Ascending/Descending Pitch Arpeggios. IEEE/ACM Trans. Audio Speech Lang. Process. **26**(11), 2194–2203 (2018). https://doi.org/10.1109/TASLP.2018.2858538

33. C. Traube, P. Depalle, in *Proc. 6th Int. Conf. on Digital Audio Effects (DAFx-03)*. Extraction of the excitation point location on a string using weighted least-square estimation of a comb filter delay (Queen Mary University of London, London, 2003)
34. H. Penttinen, V. Välimäki, A time-domain approach to estimating the plucking point of guitar tones obtained with an under-saddle pickup. Appl. Acoust. **65**(12 SPEC. ISS.), 1207–1220 (2004). https://doi.org/10.1016/j.apacoust.2004.04.008
35. Y.P. Chen, L. Su, Y.H. Yang, in *Proc. 16th Int. Society for Music Information Retrieval Conference (ISMIR)*. Electric guitar playing technique detection in real-world recordings based on F0 sequence pattern recognition (University of Malaga, Malaga, 2015), p. 708–714
36. J. Abeßer, H. Lukashevich, G. Schuller, in *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*. Feature-based extraction of plucking and expression styles of the electric bass guitar (2010), pp. 2290–2293. https://doi.org/10.1109/ICASSP.2010.5495945
37. L. Su, L.F. Yu, Y.H. Yang, in *Proc. 15th International Society for Music Information Retrieval Conference (ISMIR)*. Sparse Cepstral and Phase Codes for Guitar Playing Technique Classification (IEEE, New York, 2014), p. 9–14
38. T.W. Su, Y.P. Chen, L. Su, Y.H. Yang, TENT: Technique-Embedded Note Tracking for Real-World Guitar Solo Recordings. Trans. Int. Soc. Music Inf. Retr. **2**(1), 15–28 (2019). https://doi.org/10.5334/tismir.23
39. D. Stowell, M. Plumbley, in *Proc. Int. Computer Music Conference (ICMC)*. Adaptive whitening for improved real-time audio onset detection (Michigan Publishing, Ann Arbor, 2007), p. 312–319
40. L. Reboursière, O. Lähdeoja, R. Chesini Bose, T. Drugman, S. Dupont, C. Picard-Limpens, N. Riche, Guitar as controller. Numediart Quartely Prog. Sci. Rep. **4**(3), 41–54 (2011)
41. L. Reboursière, O. Lähdeoja, T. Drugman, S. Dupont, C. Picard, N. Riche, in *Proc. 12th Int. Conf. on New Interfaces for Musical Expression (NIME)*. Left and right-hand guitar playing techniques detection (University of Michigan, Ann Arbor, 2012), p. 1–4
42. L. Reboursière, S. Dupont, in *Intelligent Technologies for Interactive Entertainment*. EGT: Enriched guitar transcription (Springer, Cham, 2013), p. 163–168
43. A. Livshin, Automatic musical instrument recognition and related topics. Ph.D. thesis, Université Pierre et Marie Curie - Paris VI (2007)
44. J. Wilkins, P. Seetharaman, A. Wahl, B. Pardo, in *Proc. 19th Int. Society for Music Information Retrieval Conference (ISMIR)*. Vocalset: A singing voice dataset (ISMIR, Paris, 2018), p. 468–474
45. J.F. Ducher, P. Esling, in *Proc. 20th Int. Society for Music Information Retrieval Conference (ISMIR)*. Folded CQT RCNN For Real-time Recognition of Instrument Playing Techniques (2019), pp. 708–714. https://doi.org/10.5281/zenodo.3527908
46. L. Turchet, A. McPherson, M. Barthet, Co-design of a Smart Cajón. J. Audio Eng. Soc. **66**(4), 220–230 (2018)
47. B. Zamborlin, Studies on customisation-driven digital music instruments. Ph.D. thesis, Goldsmiths, University of London (2015). https://doi.org/10.25602/GOLD.00012312
48. K. Jathal, Real-time timbre classification for tabletop hand drumming. Comput. Music. J. **41**(2), 38–51 (2017). https://doi.org/10.1162/COMJ_a_00419
49. O. Lähdeoja, in *Proc. Int. Conf. on New Interfaces for Musical Expression (NIME)*. Augmenting chordophones with hybrid percussive sound possibilities (2009), pp. 102–105. https://doi.org/10.5281/zenodo.1177607
50. A. Martelloni, A. McPherson, M. Barthet, in *Proc. Int. Conf. on New Interfaces for Musical Expression (NIME)*. Percussive fingerstyle guitar through the lens of nime: an interview study (2020), pp. 440–445. https://doi.org/10.5281/zenodo.4813463
51. A. Martelloni, A. McPherson, M. Barthet, in *Proc. Int. Conf. on New Interfaces for Musical Expression (NIME)*. Guitar augmentation for percussive fingerstyle: Combining self-reflexive practice and user-centred design (2021). https://doi.org/10.21428/92fbeb44.2f6db6e6
52. A. Martelloni, A.P. McPherson, M. Barthet, in *Proc. 24th Int. Society for Music Information Retrieval Conference (ISMIR)*. Real-time percussive technique recognition and embedding learning for the acoustic guitar (2023), pp. 121–128. https://doi.org/10.5281/zenodo.10265236
53. D. Stefani, L. Turchet, in *Proc. Int. Conf. on Digital Audio Effects (DAFx20in22)*. On the Challenges of Embedded Real-Time Music Information Retrieval, vol. 3 (mdw, University of Music and Performing Arts, Vienna, 2022), p. 177–184
54. P.M. Brossier, Automatic Annotation of Musical Audio for Interactive Applications. Ph.D. thesis, Queen Mary University of London (2006)
55. D. Stefani, L. Turchet, in *Proc. 24th Int. Conf. on Digital Audio Effects (DAFx20in21)*. Bio-Inspired Optimization of Parametric Onset Detectors, vol. 2 (2021), pp. 268–275. https://doi.org/10.23919/DAFx51585.2021.9768293
56. W. Brent, in *Proc. 2010 International Computer Music Conference, ICMC*. A timbre analysis and classification toolkit for pure data (Michigan Publishing, Ann Arbor, 2010)
57. H. Ding, P.M. Feng, W. Chen, H. Lin, Identification of bacteriophage virion proteins by the ANOVA feature selection and analysis. Mol. BioSyst. **10**, 2229–2235 (2014). https://doi.org/10.1039/C4MB00316K
58. A. McPherson, R. Jack, G. Moro, in *Proc. Int. Conf. on New Interfaces for Musical Expression (NIME)*. Action-sound latency: Are our tools fast enough? (2016), pp. 20–25. https://doi.org/10.5281/zenodo.3964611
59. D. Stefani, G.A. Giudici, L. Turchet, in *Proc. 19th Int. Audio Mostly Conference*. On the importance of temporally precise onset annotations for real-time music information retrieval: Findings from the ag-pt-set dataset (2024), pp. 270–284. https://doi.org/10.1145/3678299.3678325
60. Y. Zhao, C. Wang, G. Fazekas, E. Benetos, M. Sandler, in *29th European Signal Processing Conference (EUSIPCO)*. Violinist identification based on vibrato features (2021), pp. 381–385. https://doi.org/10.23919/EUSIPCO54536.2021.9616197

Publisher's Note

Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.