OPEN ACCESS Freely available online
CC BY

N. Silva and L. Turchet, "Smart Drums: Comparing Audio and MIDI in Embedded Real-Time Drum Pattern Recognition," *J. Audio Eng. Soc.*, vol. 74, no. 11, pp. 785–795 (2026 Nov.). https://doi.org/10.17743/jaes.2026.0313.

Smart Drums: Comparing Audio and MIDI in Embedded Real-Time Drum Pattern Recognition

NISHAL SILVA,* **AND LUCA TURCHET,** *AES Fellow*
(nishal.silva@unitn.it) (luca.turchet@unitn.it)

Department of Information Engineering and Computer Science, University of Trento, Trento, Italy

This paper compares dynamic time warping (DTW) and recurrent neural networks (RNNs) for real-time drum pattern recognition on embedded hardware using both MIDI and audio representations. Motivated by Smart Drums—electronic drum kits with embedded intelligence requiring low-latency, robust pattern detection to repurpose drum phrases as high-level control signals during live performances—a dataset was recorded from 20 drummers, comprising 200 distinct drum patterns, each with ten expressive variations (2,000 performance instances), recorded in MIDI and audio. DTW serves as an alignment-based baseline using pattern-specific thresholds applied to MIDI matrices and MFCC-based audio features. The RNN employs a stacked recurrent architecture trained on synthetic pattern variations generated by a rule-based model. Experiments assess accuracy (precision, recall, and F1-score), robustness to expressive variation, and embedded performance (latency, CPU, and RAM usage) on a Raspberry Pi 4 across sets of one, three, and ten simultaneous patterns. Results show that the RNN consistently attains higher F1-scores and lower inference times than DTW in both modalities, with the performance gap widening as the number of simultaneous patterns increases. While MIDI generally yields higher detection accuracy than audio, the RNN maintains robust performance in both domains, making it better suited for deployment on Smart Drums.

Keywords: Embedded Audio, Music Information Retrieval, Music Pattern Recognition

1 INTRODUCTION

Real-time musical pattern recognition is relevant to a broad range of applications, including augmented and smart musical instruments (SMIs), audio plugins, live lighting/visual control, and interactive installations. One motivating paradigm for connected, computationally active instruments is the Internet of Musical Things, which envisions a network of smart, connected devices dedicated to the production and reception of musical content [1]. A central tenet of the Internet of Musical Things is the concept of SMIs—instruments that surpass the sole role of sound generation to become active computational devices capable of analyzing, understanding, and responding to the musician in real time [2]. Unlike traditional augmented instruments, which typically map sensor data directly to sound synthesis parameters [3], SMIs possess embedded intelligence that allows them to extract high-level semantic information from a performance. This understanding is a prerequisite for advanced forms of human-computer interaction such as automatic accompaniment, intelligent looping, context-aware effects processing, and real-time feedback.

Smart Drums represent a specific class of SMIs in which a drum kit is augmented with an embedded computing unit. In a representative use scenario, a drummer performing live repeats a recognizable pattern, such as a characteristic groove or fill, and the Smart Drum detects this repetition and transmits a high-level control signal wirelessly to external devices such as lighting controllers, smoke machines, haptic devices, loop pedals, or multimedia servers, with the aim of enhancing the audience experience. To enable a real-time operation, it is necessary to ensure that the triggered responses via the external peripherals feel musically immediate. Realizing this capability requires solving the problem of embedded real-time drum pattern recognition: identifying specific rhythmic patterns from a continuous audio or MIDI stream, on constrained hardware, under strict latency requirements.

The latency requirement between the conclusion of the pattern by the drummer and the triggering of the associated event perceived by the audience depends on the peripheral

*To whom correspondence should be addressed, email: nishal.silva@unitn.it.

involved and the corresponding sensory stimulus (visual, haptic, auditory, and olfactory). The strictest of these requirements relates to the case of triggering a sound immediately after a pattern is completed. The threshold for distinguishing two consecutive sound events is established at approximately 30 ms [4], and this value is therefore considered the upper bound on the acceptable end-to-end latency for real-time pattern classification in the Smart Drum system.

Musical pattern detection is a widely researched task with applications including computational musicology, audio fingerprinting, music classification, and music recommendation. Research has demonstrated the importance of musical patterns and their analysis, as humans usually associate structure in music through the perception of repetition [5, 6]. Real-time signal analysis has a well-established presence in music information retrieval: onset detection [7, 8], beat tracking, and real-time audio feature extraction have each been demonstrated at latencies compatible with live performance.

Real-time and streaming transcription of polyphonic and percussive audio has also seen substantial progress [9, 10] and can in principle support pattern-level applications since transcribed onsets/pitches can be matched against a template as they arrive. That broader body of work targets open-set, frame-by-frame transcription of arbitrary musical content; the present work instead addresses closed-set recognition of a fixed, drummer-specific pattern vocabulary, complete only once a full pattern has been played, which is a distinct task with different constraints. Work of this kind is regularly disseminated at international conferences such as New Interfaces for Musical Expression, International Society for Music Information Retrieval, and Sound and Music Computing, which serve as the principal platforms for presenting new real-time and interactive music systems to the computer music community.

The specific problem of detecting user-defined, high-level percussive patterns in real-time on embedded hardware, under the narrower setting of closed-set pattern recognition for SMIs, has not been systematically investigated. This gap currently limits the development of Smart Drums as an SMI platform.

Furthermore, the computational constraints of SMIs impose strict limitations on algorithmic complexity. Traditional methods may struggle with real-time operation and polyphony [11–13], and cloud-based solutions may exceed the strict latency threshold set for real-time operation (i.e., 30 ms). Consequently, recognition algorithms must be optimized for edge execution: operations running locally on embedded microprocessors where memory and clock cycles are scarce. To address these challenges, this research evaluates two distinct approaches: a traditional dynamic time warping (DTW) approach and a data-driven recurrent neural network (RNN)–based approach.

DTW is a widely used technique for sequence alignment in music information retrieval. By computing a nonlinear alignment path between a query sequence and a reference template, DTW can provide an elastic measure of similarity that is robust to global tempo changes and local timing distortions. However, the standard DTW algorithm suffers from quadratic computational complexity ($O(N^2)$), which scales poorly with sequence length and can become a bottleneck in real-time systems. Moreover, DTW often struggles with raw audio data, where noise and timbral variations can corrupt the distance metric, necessitating complex and computationally expensive feature extraction steps.

In contrast, deep learning architectures such as RNN, with their ability to handle long sequential events, have emerged as powerful alternatives for sequence modeling tasks [14, 15]. RNNs are capable of learning high-dimensional, nonlinear representations of temporal data, potentially allowing them to capture subtle performance deviations more effectively than rigid distance metrics. Moreover, once trained, RNN typically offer lower inference complexity ($O(N)$), making them a strong candidate for real-time embedded applications. However, deep learning models are notoriously data-hungry and can function as black boxes, making their failure modes difficult to predict compared to the interpretable alignment paths of DTW.

While existing literature has explored both methods in isolation, there is a lack of rigorous comparative analysis focusing specifically on retrieval accuracy under conditions of high expressive variation in real-time contexts. Furthermore, few studies explicitly compare the performance of these architectures across both symbolic (MIDI) and acoustic (audio) domains, a distinction that is critical for SMIs, which may rely on either piezoelectric triggers (MIDI-like) or internal microphones (audio).

This paper addresses this gap by presenting a comprehensive comparative study of DTW and RNNs for drum pattern detection. To ensure the ecological validity of this evaluation, a specialized dataset comprising 200 distinct drum patterns was constructed. For each pattern, professional drummers recorded ten expressive variations, deliberately applying artistic expressive techniques including micro-timing shifts, ghost notes, and dynamic accents. The resulting dataset comprises a total of 2,000 performance instances, each recorded simultaneously in MIDI and audio, and serves as the ground truth for this evaluation.

The investigation is structured around two primary experiments: (1) a comparison of DTW and RNN performance on symbolic MIDI data and (2) a parallel comparison on raw audio data. By isolating these modalities, the aim is to disentangle the algorithmic strengths of each method from the challenges of feature extraction. Experiments covering pattern sets of one, three, and ten simultaneous patterns evaluate accuracy, robustness to expressive variation, and embedded performance on a Raspberry Pi 4. The results of this study are intended to provide concrete architectural recommendations for the design of next-generation SMIs, in particular Smart Drums, identifying the candidate that best balances the conflicting demands of accuracy, expressivity, and real-time efficiency.

2 RELATED WORKS

A wide-ranging survey of pattern discovery in music is presented in [16], covering computational approaches to

identifying repeated musical patterns. The authors highlighted challenges related to method comparison, music representation, and evaluation against patterns perceived by human listeners, advocating closer integration of computational and perceptual approaches.

The work in [17] proposed an offline technique for uncovering repetitive note patterns in polyphonic music through segmentation-based analysis. Built upon the Structural Features algorithm, the method detects repeated sequences by examining pitch and rhythmic content, yielding strong results in identifying both exact and approximate repetitions within complex polyphonic material.

Novel pattern-matching algorithms for symbolically encoded polyphonic music were introduced in [18], with a particular focus on robustness to transposition. The proposed methods outperformed earlier approaches in both accuracy and computational efficiency when applied to complex polyphonic structures.

In [19], overlapping sound events in polyphonic audio were tackled using non-negative matrix factorization to construct dictionaries from annotated recordings. The resulting method demonstrated a strong capability to separate and identify concurrent sound events.

The authors of [11] tackled the problem of locating repeated note content directly within polyphonic audio, circumventing the need for symbolic representations. This work marked a meaningful step forward in purely audio-driven approaches to music pattern analysis.

The work of [20] addressed finding repetitions within intricate polyphonic textures, simultaneously accounting for pitch, rhythm, and timbre, enabling structural analysis of multivoice polyphonic compositions. The authors of [21] proposed an adaptive redundancy filtering approach for extracting motivic patterns from polyphonic music, enabling the identification of salient musical motifs embedded within complex multivoice textures.

Collectively, polyphonic audio pattern recognition has been approached through a diverse range of signal processing and machine learning paradigms. However, these contributions target offline settings (i.e., finding all occurrences of a given musical pattern in a musical score or recording), whereas the present work investigates real-time detection from a continuous stream.

The authors' previous studies investigated real-time musical pattern detection for audio and MIDI, respectively. The work presented in [22] compared DTW and RNN approaches for detecting symbolic monophonic patterns from MIDI, finding that while both methods performed comparably for up to one distinct pattern, DTW accuracy degraded significantly as the number of patterns increased to ten, whereas the RNN maintained a detection accuracy of approximately 98%. Such a work was later extended to polyphonic audio [23]. It was demonstrated that the RNN achieved a superior F1-score of 0.76 compared to 0.655 for DTW when detecting a single pattern, with an average inference latency of 14.69 ms versus 1,038.34 ms for DTW on an embedded Raspberry Pi 4, confirming its suitability for real-time deployment in SMIs. However, such studies did not involve percussive sounds.

Techniques for processing audio in real-time with minimal latency have been advancing steadily, enabling tighter integration with live musical contexts [24, 25]. Achieving this typically demands careful algorithmic optimization alongside dedicated hardware and real-time operating environments [26, 27]. Such infrastructure is foundational for SMI development, underpinning use cases from live performance to interactive installations and real-time effects. However, the specific challenge of embedding real-time polyphonic pattern detection within an SMI remains largely unaddressed, along with defining interaction paradigms that exploit this capability. The present work aims to bridge this gap.

3 DATASET OF DRUM PATTERNS

A dedicated dataset of expressive drum performances is essential to evaluate any percussive pattern detection method. Such a method must be able to recognize patterns not only under ideal, quantized conditions, but also in realistic performance scenarios where micro-timing deviations, dynamics, and timbral variations are present. Without a controlled yet expressive corpus, it is impossible to compare traditional alignment-based methods and data-driven models across symbolic and audio domains, while ensuring that the accuracy differences are due to algorithmic properties rather than uncontrolled variability in the input.

A custom dataset was created to facilitate a fair comparison between the two methods. The dataset was created with the contributions of 20 drummers: all Italian males aged between 18 and 36 (mean age: 26.0, SD: 5.07), at least 5 years of performing/recording/composing experience. Each drummer contributed ten unique drum patterns, yielding 200 distinct drum patterns in total (identifiable by unique *artistID–patternID* pairs, such that same *patternID* values across different *artistID*s are distinct musical phrases). The patterns represent musically meaningful grooves and motifs, a range of metric structures (e.g., straight and swung feels, binary and ternary subdivisions), stylistic idioms (e.g., rock, funk, Latin-inspired patterns), and levels of rhythmic density. For each of the 200 patterns, the drummers were asked to record ten deliberate expressive variations, yielding a total of 2,000 performance instances. These variations were designed to preserve the perceptual rhythmic identity while introducing realistic modifications such as micro-timing shifts (ahead/behind the beat), dynamic phrasing (accent patterns, ghost notes, crescendi/decrescendi), and subtle orchestrational changes within the drum kit (e.g., occasional hi-hat openings or alternative cymbal choices).

A single electronic drum kit (Yamaha DTX-432k) was used to record the entire dataset to ensure that deviations are exclusively due to the drummers and not the equipment. The choice of an electronic drum kit was due to the ease of obtaining MIDI data. Its silent operation was also a crucial factor when conducting experiments in laboratory conditions, as well as its small footprint and ease of transportation. The dataset was recorded in both MIDI and audio. The MIDI input was obtained using the electronic

drum kit's MIDI interface, and the mono audio was captured through a high-fidelity audio interface (MOTU M2) in 48 kHz, 24-bit to ensure sufficient temporal and spectral detail for feature extraction.

The dataset follows the same framework as the present authors' previous datasets of monophonic MIDI patterns [22][1] and polyphonic audio patterns [23][2] recorded using the electric guitar and keyboard. Thus, the file structure was kept consistent as the previous datasets. Each file is named $artistID_patternID_versionID.wav$ and $artistID_patternID_versionID.mid$, where $\mathbb{Z}$ represents a set of integers and $artistID \in \mathbb{Z} : 40 \leq artistID \leq 59$, $patternID \in \mathbb{Z} : 0 \leq patternID \leq 9$, and $versionID \in \mathbb{Z} : 0 \leq versionID \leq 9$. *artistID* represents the musician, *patternID* represents the different patterns. *versionID* represents the versions of the patterns, where $versionID = 0$ is the ground truth, and each subsequent iteration is a repetition with expressive techniques. The dataset is publicly available online in two parts.[3,4]

The dataset was constructed with the sole intent of capturing a large number of expressive variations performed by multiple musicians. The large number of variations is a crucial aspect when training deep learning–based solutions. Each variation was ensured to be perceptually equivalent to the ground truth. Upon analyzing the dataset, some of the most common expressive variations were identified:

- *Dynamic variations:* Ghost notes and accents, crescendi and diminuendi within a pattern, sudden velocity changes, cross-stick and rim-shot variations, and switching between left and right hand.
- *Temporal variations:* Micro-timing adjustments, push-pull techniques, subdivision variations, tempo changes, and single, double, and half-time techniques.
- *Embellishments:* Variations in drum fills, grace notes, interchanging toms within a pattern, addition or removal of notes, and inclusion of drum rolls.
- *Stylistic variations:* Genre-specific groove changes, ethnic and cultural rhythm adaptations, and fusion of different styles.

4 METHODOLOGY

4.1 System Architecture

The proposed system is designed to enable real-time detection of drum patterns directly on an embedded platform integrated into an electronic drum kit or hybrid acoustic–electronic setup, resulting in a Smart Drum system. The embedded computing unit is a Raspberry Pi 4, chosen for its balance between processing power, energy consumption, and availability of general-purpose input/output and network interfaces suitable for SMIs. The instrument's audio and MIDI outputs are split and routed both to the musician's signal path (amplifier, audio interface, or PA) and to the embedded system, ensuring that pattern detection operates noninvasively on a parallel signal path. Fig. 1 shows a block diagram of the system architecture.

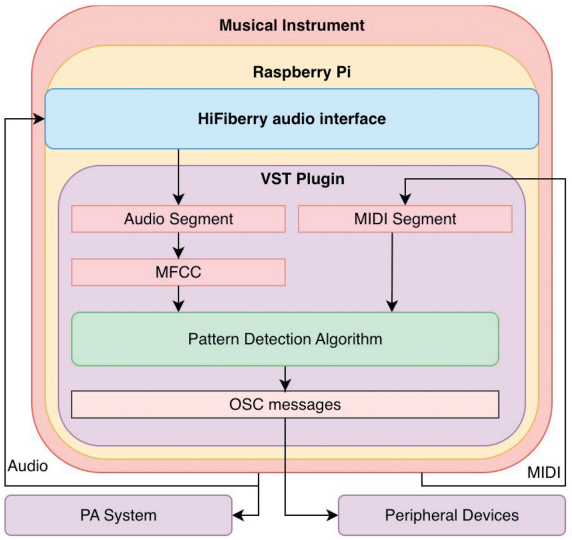


Fig. 1. A block diagram of the hardware and software setup of the system embedded in the musical instruments.

On the embedded side, the audio is captured through a HiFiBerry audio interface, while the MIDI is received directly from the electronic drum kit. The system runs ELK Audio OS, a low-latency Linux distribution providing a real-time kernel and optimized audio drivers to keep end-to-end latency below the latency required for live performance [28]. The pattern detection logic is encapsulated within a virtual studio technology plugin that handles audio/MIDI input, feature extraction, classification, and control message generation.

For both the DTW and RNN approaches, incoming signals are processed in overlapping windows. For audio, a sliding window of a length matching or slightly exceeding the duration of the target pattern is extracted every 30 ms, a hop size previously used to match the temporal resolution of human auditory discrimination [4] and to guarantee that processing finishes before the next frame arrives. For MIDI, an analogous symbolic window is maintained via a rolling buffer of recent events; its length set to accommodate the longest drum pattern plus a margin for added notes introduced by expressive variation, following the strategy adopted in the present authors' prior symbolic pattern detection work [22].

Upon detecting a pattern, the embedded system transmits Open Sound Control messages wirelessly via Wi-Fi to external devices (e.g., lighting controllers, media servers, and haptic wearables), reusing the same communication layer previously demonstrated for smart electric guitar and keyboard prototypes. This architecture supports concurrent deployment of both DTW and RNN pipelines during evaluation, with shared feature extraction but separate decision

[1] https://doi.org/10.5281/zenodo.10818616.
[2] https://doi.org/10.5281/zenodo.14497998.
[3] DoMP: https://doi.org/10.5281/zenodo.14497974.
[4] DoMP2: https://doi.org/10.5281/zenodo.18395007.

blocks, and is directly aligned with earlier SMI designs that were successfully deployed and tested in live contexts [29].

4.2 Dynamic Time Warping

DTW is employed as a traditional benchmark for measuring similarity between an incoming performance and stored reference patterns [30]. DTW computes an optimal alignment path between two temporal sequences that may differ in length and local speed, yielding a distance measure that is robust to tempo fluctuations and small timing deviations. This property makes DTW conceptually well suited for drum pattern detection, where expressive micro-timing and slight structural variations occur between repetitions. However, DTW's quadratic complexity in sequence length and its dependence on carefully chosen feature representations impose practical constraints for real-time deployment on embedded devices.

4.2.1 MIDI-Based Dynamic Time Warping

For MIDI-based DTW, the method extends the symbolic monophonic representation developed in prior work on guitar and keyboard patterns to the drum domain, adapting it to percussion-specific event types. The MIDI stream from the drum kit is first parsed into a sequence of events, where each event corresponds to a note onset (drum hit) or a rest, with associated attributes such as MIDI pitch (drum instrument), onset time, and duration. Velocity is excluded from the similarity computation, as earlier analyses on expressive monophonic datasets showed that velocity varies substantially even between repetitions of the same pattern and degrades DTW-based similarity when included [22].

Each drum pattern is encoded as a two-dimensional sequence with one dimension representing pitch and the other representing note durations, forming a pitch–duration matrix analogous to that used in symbolic monophonic pattern detection. To accommodate expressive variations—such as added ghost notes or omitted hits—the DTW window size is chosen to be approximately 20%–25% longer than the nominal pattern length, following the empirically derived margin used in previous work where musicians frequently added up to approximately one quarter of the pattern's note count as extras [22]. During online operation, a rolling MIDI window is maintained, converted to this matrix representation, and compared against each stored pattern.

A multidimensional DTW is then computed between the pitch–duration matrix of the incoming window and that of each reference pattern. As in earlier work, a merged-path strategy is adopted: separate one-dimensional DTWs are computed for pitch and duration, and their distances are combined via a weighted sum, allowing pitch to contribute more strongly to the final score than duration, in line with empirical findings that pitch is more decisive for perceived pattern identity [31]. The Euclidean distance is used as the base local distance metric between corresponding feature vectors. For each pattern, a pattern-specific detection threshold is obtained from a held-out validation set of drum recordings by maximizing the F1-score using a Youden-type optimization procedure [32], mirroring the threshold calibration procedure used for polyphonic audio pattern detection on guitar and keyboard. At runtime, if the DTW distance between the current window and a reference pattern falls below the learned threshold, a detection is triggered and the corresponding Open Sound Control message is sent.

4.2.2 Audio-Based Dynamic Time Warping

For audio-based DTW, the system follows the polyphonic audio methodology previously developed for smart guitar and keyboard, adapted to mono drum recordings. The drum audio is first segmented into sliding windows of a length corresponding to the nominal pattern duration, extracted every 30 ms. For each window, Mel frequency cepstral coefficients (MFCCs) are computed using an FFT size of 2,048 samples and a hop size of 512 samples, producing overlapping spectral frames that capture both rhythmic and timbral characteristics of the drum performance while maintaining a compact representation suitable for embedded computation. A fixed number of MFCC coefficients (e.g., ten per frame) is retained, consistent with previous experiments [23] where this configuration offered a good compromise between accuracy and speed on the Raspberry Pi 4.

Each window is therefore represented as a 2D MFCC spectrogram (time frames $\times$ coefficients). Multidimensional DTW is applied between this spectrogram and the stored MFCC spectrograms of each ground-truth drum pattern, effectively aligning time–frequency representations. As in the MIDI case, DTW distances are computed for each pattern, and pattern-specific thresholds are estimated using a subset of recorded drum variations, ensuring that the method tolerates realistic changes in timing and dynamics while still discriminating between distinct patterns. The same hop size and windowing strategy are used during evaluation as in the guitar/keyboard audio system to allow direct comparison of latency and computational load.

4.3 Recurrent Neural Network

RNNs are employed as the deep learning–based alternative to DTW, building on prior work where stacked RNN architectures (combining SimpleRNN, LSTM, and GRU layers) achieved state-of-the-art performance for real-time pattern detection on symbolic monophonic MIDI and polyphonic audio. RNNs are well suited for sequential data because they are able to model long-term temporal dependencies and nonlinear relationships, which are crucial for capturing expressive drum performance characteristics such as groove, timing deviations, and dynamic accents. In earlier SMI applications, such architectures showed both high accuracy and low inference latency (the time required to compute a single forward pass through the network for one analysis window) when deployed on a Raspberry Pi 4, making them strong candidates for real-time embedded drum pattern detection.

4.3.1 MIDI-Based Recurrent Neural Network

For the MIDI-based RNN, the input representation extends the discrete-time encoding used for guitar and keyboard to percussive MIDI streams. The incoming MIDI

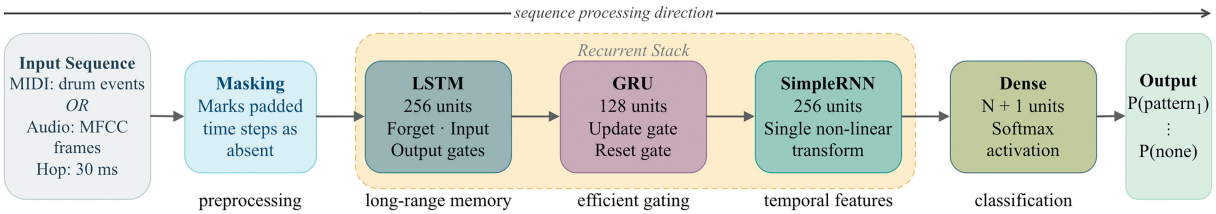


Fig. 2. Stacked RNN architecture for drum pattern detection. A masking layer is followed by LSTM (256 units), GRU (128 units), and SimpleRNN (256 units) layers, culminating in a dense softmax output over N pattern classes plus *no-pattern*. The architecture is shared across MIDI or audio conditions.

events (drum hits and rests) are sampled at fixed 30 ms intervals, producing a time series in which each time step encodes the state of the drum kit at that instant. At each sample, the representation comprises a pitch-like index encoding the active drum instrument and temporal information such as interonset intervals embedded implicitly through the sampling grid. As in the previous symbolic study, velocity is excluded due to its high variability across repetitions and its negative impact on learning stable pattern embeddings.

To enable RNN training, each ground-truth drum pattern is resampled into this discrete time-series format, yielding sequences of a length dependent on pattern duration and 30 ms hop size. Since training requires fixed-size sequences, all pattern sequences are zero-padded to match the longest pattern in the training set, and a masking layer in the network is used to ignore padded time steps during learning. The masking layer is used solely to standardize sequence length, such that the recurrent layers treat them as absent rather than as zero-valued inputs. The RNN is trained as a sequence classifier with one output class per drum pattern plus an additional no-pattern class for background activity, mirroring the multiclass setup used previously for groups of monophonic patterns [31].

As in the guitar/keyboard study, the training corpus is augmented synthetically using a rule-based model of expressive variation. The rules are adapted from the monophonic symbolic work and extended to percussive patterns, incorporating operations such as adding or removing drum hits, altering interonset timing within bounds, inserting ghost notes, and making small structural changes while preserving perceptual pattern identity. These rules are derived from empirical analyses of expressive variations in previous datasets [22, 23], where typical ranges for added/removed notes, timing deviations, and additional pauses were quantified. For each drum pattern, a large number (on the order of several thousand) of synthetic variations are generated, producing a training corpus that reflects the range of expressivity expected in live drumming while limiting the burden on musicians to record many repetitions.

The RNN architecture follows the stacked configuration previously validated on symbolic monophonic data: a masking layer, followed by a long short-term memory (LSTM) layer [14] (256 units), a gated recurrent unit (GRU) layer [15] (128 units), and a simple recurrent network (SimpleRNN) layer (256 units), culminating in a dense output layer with a softmax activation over the pattern classes plus no-pattern (see Fig. 2). LSTM uses gated memory cells to capture long-range dependencies while mitigating the vanishing gradient problem [14]; GRU is a computationally lighter variant that merges forget and input gates into a single update gate [15]; SimpleRNN applies a single nonlinear transformation of the concatenated input and previous hidden state. Softmax normalizes the output logits into a probability distribution summing to one, such that each entry represents the model's confidence that the current window contains the corresponding pattern. The model is trained using cross-entropy loss and an Adam optimizer [33], which adapts the per-parameter learning rate using first and second-order gradient moment estimates, with early stopping based on validation F1-score to prevent overfitting. This configuration is chosen because earlier experiments [22, 23] showed a good trade-off between accuracy and complexity when deployed on embedded hardware.

4.3.2 Audio-Based Recurrent Neural Network

For the audio-based RNN, the input representation is aligned with the polyphonic audio methodology for smart guitar and keyboard, now applied to drum recordings. The incoming audio stream is segmented into sliding windows extracted every 30 ms, identical to the DTW pipeline. For each window, MFCCs are computed with an FFT size of 2,048 and a hop size of 512, producing a sequence of MFCC frames over the pattern's duration. Each frame contains a fixed number of coefficients (e.g., ten), yielding a *time* × *features* matrix per window.

Before training, all MFCC sequences are zero-padded to a uniform temporal length equal to the longest drum pattern in the dataset, and a masking layer is used to exclude padded frames. The stacked RNN architecture (*Masking* → *LSTM* → *GRU* → *SimpleRNN* → *Dense*) is reused, taking MFCC sequences as input and outputting class probabilities over the drum patterns plus "no-pattern." The same synthetic-variation strategy used for MIDI is applied at the feature level: rule-based transformations are directly applied to the MFCC spectrograms of each ground-truth drum pattern to generate a large synthetic training corpus, using operations such as temporal stretching/compression, amplitude scaling, frame duplication/removal, and additive noise, as in the audio-based system developed for guitar and keyboard.

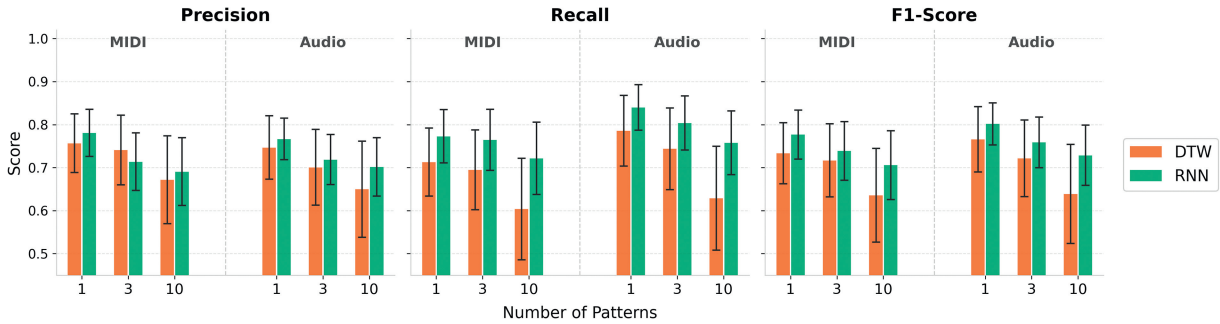


Fig. 3. Precision, recall, and F1-score (mean ± 1 SD) for DTW and RNN across MIDI and audio modalities at pattern set sizes of one, three, and ten.

This approach avoids the need to synthesize new audio and instead produces feature-space variations that mimic plausible expressive changes observed in the recorded dataset.

5 EVALUATION

Experiments were structured to mirror the evaluation protocols from the authors' prior symbolic and audio studies to the drum context, covering pattern sets of one, three, and ten. For each condition, independent DTW and RNN models were configured for the MIDI and audio modalities. For DTW, each drum pattern's ground-truth sequence served as the reference, and its nine expressive variations were treated as positive examples, while all other patterns and their variations served as negatives. Each incoming MIDI sequence window is compared via DTW to all reference drum patterns; a detection is considered correct if the minimal distance falls below the corresponding pattern's threshold and the predicted pattern ID matches the ground truth. For the RNN, models were trained exclusively on synthetic variations of the ground-truth patterns (MIDI or MFCCs) and evaluated on real drum performances from the recorded dataset, ensuring that test data reflects realistic expressive variation rather than synthetic transformations. For each sliding window, the RNN outputs a probability distribution over classes; a pattern is considered detected if its predicted probability exceeds a fixed threshold and the most probable class corresponds to the true pattern ID within the window. As with DTW, this threshold is determined on the same held-out validation set, selecting the value that maximizes F1-score.

Detection performance is quantified using standard evaluation metrics—precision, recall, and F1-score. Precision measures the proportion of triggered detections that correspond to a true pattern occurrence; recall measures the proportion of true pattern occurrences that are detected; and the F1-score, defined as $F1 = 2 \cdot \frac{P \cdot R}{P+R}$, is their harmonic mean, ranging from 0 (worst) to 1 (best). All metrics are computed across all drum patterns and variations, with mean and SD reported across pattern groups. Embedded performance (detection time, CPU usage, and RAM usage) is measured on the Raspberry Pi 4 under ELK Audio OS, following the protocol from prior embedded evaluations [23].

6 RESULTS

Across all 200 drum patterns and their expressive variations, both DTW and RNN achieved reliable detection of repeated patterns in MIDI and audio, but with markedly different error profiles. Tables 1, 2, and 3 report precision, recall, and F1-score for pattern sets of one, three, and ten patterns, respectively. Fig. 3 reports all results (mean ± SD across pattern groups). Table 4 reports average computational performance on the Raspberry Pi 4.

In the audio domain, the RNN achieved a higher F1-score than DTW across all conditions, in line with previous polyphonic pattern detection experiments [24]. For drum audio, DTW achieved higher precision in only a few cases, indicating fewer false positives when a detection occurred, whereas the RNN showed higher recall in a majority of the conditions, capturing a larger proportion of true pat-

Table 1. Comparison of RNN and DTW methods in detecting a single pattern.

	Precision	Recall	F1 Score
DTW—MIDI	0.757	0.713	0.734
DTW—Audio	0.747	0.786	0.766
RNN—MIDI	0.781	0.773	0.777
RNN—Audio	0.767	0.840	0.802

Table 2. Comparison of RNN and DTW methods in detecting three patterns.

	Precision	Recall	F1 Score
DTW—MIDI	0.741	0.695	0.717
DTW—Audio	0.701	0.744	0.722
RNN—MIDI	0.714	0.765	0.739
RNN—Audio	0.719	0.804	0.759

Table 3. Comparison of RNN and DTW methods in detecting ten patterns.

	Precision	Recall	F1 Score
DTW—MIDI	0.672	0.604	0.636
DTW—Audio	0.650	0.629	0.639
RNN—MIDI	0.691	0.722	0.706
RNN—Audio	0.702	0.758	0.729

Table 4. Average pattern detection time, CPU usage, and RAM usage on Raspberry Pi 4 for MIDI and audio conditions.

	MIDI		Audio	
	RNN	DTW	RNN	DTW
Detection time (ms)	15.76	365.6	21.43	1132.2
CPU usage (%)	35.5	42.86	34.6	46.8
RAM usage (MB)	489.23	345.67	532.76	243.64

tern occurrences but with more occasional confusions between rhythmically similar patterns. This trade-off matches earlier observations on guitar and keyboard, where DTW's strict template matching favored precision, while the RNN's learned invariances favored recall under expressive variation.

In the MIDI domain, the RNN similarly outperformed DTW in F1-score across all conditions. The advantage grew with pattern count: at one pattern, the RNN-DTW F1 gap was approximately 0.04 (MIDI) and 0.04 (audio); at ten patterns, this gap widened to approximately 0.07 (MIDI) and 0.09 (audio). This replicates the finding from the symbolic monophonic study [23] in which DTW degraded significantly beyond three to five patterns while the RNN maintained high accuracy. The SD of DTW's F1-score widened substantially from one to ten patterns (MIDI: 0.071 to 0.109; audio: 0.076 to 0.115), indicating increasing inconsistency across different pattern groups. The RNN's SD grew more modestly (MIDI: 0.057 to 0.080; audio: 0.049 to 0.070), confirming greater reliability across diverse pattern sets.

The experiment confirmed that expressive performance challenges both methods differently. As in the guitar and keyboard datasets, drummers introduced timing deviations, added or removed hits, ghost notes, and dynamic accents that preserved perceptual pattern identity while breaking strict one-to-one correspondence. DTW's accuracy remained relatively stable with respect to pattern length, reflecting earlier observations that alignment-based matching is largely insensitive to sequence length within the tested range, provided that thresholds are well calibrated. In contrast, the RNN's performance exhibited a mild decline for the longest drum patterns, paralleling the prior polyphonic audio study [24] where longer patterns introduced more opportunities for variation and accumulated prediction errors.

Despite this, the RNN showed superior robustness to expressive variation overall. In these drum experiments, as in the monophonic symbolic study [23], DTW distances increased markedly for variations with added or removed events, producing false negatives unless thresholds were relaxed, which then led to more cross-pattern confusions. The RNN, trained on synthetic variations derived from rules calibrated on previous pattern datasets, generalized better to the drummers' expressive changes, yielding higher recall and more consistent performance across musicians. This confirms that the rule-based augmentation strategy originally designed for melodic keyboard and guitar patterns also transfers effectively to rhythmic drum material.

For the audio domain specifically, the MFCC-based representation implicitly encodes dynamics: louder playing shifts the spectral content, so timbral changes due to playing intensity are captured in the feature representation and handled by both methods. In the MIDI domain, velocity was explicitly excluded from the feature representation, so both DTW and RNN operate purely on pitch and timing information; the robustness observed in the MIDI experiments therefore reflects tolerance to timing and structural variation rather than dynamic variation.

As shown by Table 4, average computational performance was assessed on a Raspberry Pi 4 to evaluate the suitability of both methods for embedded real-time drum pattern detection, following the same measurement protocol adopted in previous smart instrument studies [24]. Consistent with earlier results on guitar and keyboard, the RNN-based approach achieved inference times on the order of a few tens of milliseconds per analysis window, well below the 30-ms hop size used for streaming operation, whereas the DTW-based approach exhibited latencies that were one to two orders of magnitude higher due to its quadratic complexity in sequence length. This difference in latency was mirrored in CPU usage: the RNN required only a moderate share of the available processing resources, enabling continuous real-time operation alongside the audio stack, while DTW consumed substantially more CPU, making it difficult to guarantee stable performance under live conditions. Memory consumption showed the opposite trend, with DTW requiring less RAM than the RNN due to its need to store model weights and recurrent states, but the observed RAM footprint of the RNN remained within acceptable limits for the target embedded platform.

7 DISCUSSION

The results confirm that both DTW and RNN-based approaches can be used to detect expressive drum patterns, yet they exhibit markedly different performance characteristics in terms of accuracy, robustness, and embedded feasibility. The central finding is that while the two methods achieve comparable F1-scores at small pattern set sizes (one to three patterns), the gap widens substantially at ten patterns, and the computational cost difference makes RNN the feasible option for real-time embedded deployment regardless of pattern set size.

From an accuracy standpoint, the drum experiments reproduce the trade off previously observed in polyphonic audio pattern detection [24]: DTW tends to yield higher precision but lower recall in some limited cases, while the RNN maintains a higher overall F1-score. This behavior can be attributed to the fundamental nature of the two approaches. DTW relies on direct distance computations between the incoming sequence and stored templates, making it sensitive to added or removed events and to certain types of timing deformation, even when these remain perceptually consistent with the underlying pattern. The RNN, trained on a large synthetic corpus of variations derived from rules calibrated on data, adapts to tolerate typical ex-

pressive changes without requiring explicit modeling of all possible deviations. This is particularly evident for longer drum patterns, where DTW's performance remains stable but the RNN's accuracy exhibits only a moderate decrease, despite the greater opportunity for expressive divergence across the pattern's duration.

The comparison between symbolic and audio domains shows the continued advantage of working in the MIDI domain when reliable symbolic information is available but also highlights the strengths of the RNN approach in handling audio directly. For MIDI, both methods achieve higher precision than in the audio case, reflecting on the absence of timbral variability and recording artifacts. However, consistent with previous monophonic results, the RNN maintains high detection accuracy even as the number of patterns in the model increases, whereas DTW degrades as each additional pattern introduces another full distance computation and a potential source of confusion.

In the audio domain, the RNN still provides a clear advantage by operating on MFCC-based representations and being trained on spectrogram-level synthetic variations. The RNN shows a high tolerance to differences in drum hits, dynamics, micro timing variations, and playing intensity, mirroring the behavior reported for polyphonic guitar and keyboard recordings. DTW remains competitive for small pattern sets and short windows but is more affected by timbral variations and background noise, which push distances beyond detection thresholds unless those thresholds are relaxed in ways that increase false positives.

A primary objective of this research was to assess the suitability of each method for deployment in embedded environments. In this regard, the RNN-based approach clearly emerges as the more favorable solution. The average inference time remains well below the 30-ms threshold. DTW, on the other hand, exhibits computation times that are one to two orders of magnitude higher under comparable conditions, due to the quadratic growth of its distance matrix with sequence length and the need to compare each window against all stored patterns. As a result, DTW cannot feasibly operate at the same window rate on the embedded platform without either drastically reducing the pattern set or compromising latency requirements.

The resource usage further clarifies these trade-offs. The RNN consistently requires less CPU during classification, thereby reducing the risk of audio glitches or dropped frames and making it more attractive for integration into SMIs that must share processing resources between synthesis, effects, and networking. DTW retains an advantage in RAM usage, thanks to its relatively small memory footprint compared to the need to store model weights and recurrent states for the RNN. However, given the available memory on contemporary embedded boards and the modest size of the models required for practical pattern sets, this difference is unlikely to be a limiting factor in typical SMI scenarios.

Taken together, these observations suggest that while DTW remains valuable as an analytical and baseline tool, the RNN-based method offers a more promising solution for real-time drum pattern detection in SMIs, especially when considering the combined requirements of accuracy, robustness, latency, and computational efficiency.

8 CONCLUSION

This work presented a systematic comparison between a DTW-based benchmark and an RNN-based method for real-time detection of expressive drum patterns in both MIDI and audio domains. The study extends previous efforts on guitar and keyboard by introducing a dedicated drum dataset comprising 200 patterns, each with ten expressive variations, and by evaluating both methods across symbolic and audio signal representations for pattern sets of one, three, and ten simultaneous patterns under conditions that emulate live performance. The comparison between the two modalities showed that MIDI consistently provides higher detection accuracy than audio, reflecting the absence of timbral variability and recording artifacts. Nevertheless, the RNN maintained robust performance in both domains, demonstrating that accurate pattern recognition is also feasible directly from audio signals.

The results demonstrate that although DTW remains a useful reference, the RNN-based approach provides superior overall performance with a faster detection time for the targeted application. Specifically, the RNN achieves higher F1-scores by combining a high recall with adequate precision, exhibits greater robustness to expressive timing and dynamic variation, and satisfies the strict latency and CPU constraints associated with embedded real-time operation. The advantage of the RNN over DTW was consistently observed in both MIDI and audio domains and became more pronounced as the number of simultaneously recognized patterns increased.

These findings support the adoption of RNN-based classifiers as the primary pattern recognition engine for real-time applications in next-generation smart drum instruments. When symbolic information is available, MIDI remains the preferred representation owing to its higher detection accuracy. However, the strong results obtained directly from audio indicate that the proposed approach is also well suited to acoustic or hybrid drum setups where MIDI is unavailable or impractical. Beyond Smart Drums, the proposed approach is equally applicable to DAW plugins, live lighting/show control, education and practice tools, and other New Interfaces for Musical Expression, International Society for Music Information Retrieval, and Sound and Music Computing–adjacent applications that rely on live pattern detection as a control modality.

Various avenues for future work emerge from the present study. First, while the rule-based synthetic augmentation is effective in capturing a broad range of expressive variations, further refinements could be explored, including data-driven augmentation schemes or adversarial training paradigms that better model rare or idiosyncratic performance behaviors. Second, the current evaluation focuses on within-instrument, within-drummer conditions. Extending the experiments to cross-drummer and cross-kit scenarios would offer insights into the models' ability to generalize across playing styles and instrument configurations. Third,

integrating the proposed drum pattern detection system into full SMI prototypes opens up opportunities for user studies that investigate how drummers would use pattern-based control in practice.

9 ACKNOWLEDGMENT

The authors acknowledge the support of the MUSMET project funded by the EIC Pathfinder Open scheme of the European Union (Grant No. 101184379). Views and opinions expressed are however those of the authors only and do not necessarily reflect those of the European Union or the European Innovation Council.

10 REFERENCES

[1] L. Turchet, C. Fischione, G. Essl, D. Keller, and M. Barthet, "Internet of Musical Things: Vision and Challenges," *IEEE Access*, vol. 6, pp. 61994–62017 (2018 Sep.), https://doi.org/10.1109/ACCESS.2018.2872625.

[2] L. Turchet, "Smart Musical Instruments: Vision, Design Principles, and Future Directions," *IEEE Access*, vol. 7, pp. 8944–8963 (2018 Oct.), https://doi.org/10.1109/ACCESS.2018.2876891.

[3] A. McPherson, "Buttons, Handles, and Keys: Advances in Continuous-Control Keyboard Instruments," *Comput. Music J.*, vol. 39, no. 2, pp. 28–46 (2015 Jun.), https://doi.org/10.1162/COMJ_a_00297.

[4] B. Moore, *An Introduction to the Psychology of Hearing* (Brill, Leiden, the Netherlands, 2012).

[5] R. Dannenberg and N. Hu, "Pattern Discovery Techniques for Music Audio," in Proc. *3rd Int. Conf. Music Information Retrieval (ISMIR)*, pp. 63–70, Paris, France (2002 Oct.).

[6] I. Y. Ren, H. V. Koops, A. Volk, and W. Swierstra, "In Search of the Consensus Among Musical Pattern Discovery Algorithms," in Proc. *18th Int. Society for Music Information Retrieval Conf.*, pp. 671–678, Suzhou, China (2017 Oct.).

[7] S. Dixon, "Onset Detection Revisited," in Proc. *9th Int. Conf. Digital Audio Effects (DAFx)*, pp. 133–137, Boston, MA, USA (2006 Sep.).

[8] S. Böck, F. Krebs, and M. Schedl, "Evaluating the Online Capabilities of Onset Detection Methods," in Proc. *13th Int. Society for Music Information Retrieval Conf. (ISMIR)*, pp. 49–54, Porto, Portugal (2012 Oct.).

[9] P. Weber, C. Uhle, M. Müller, and M. Lang, "Real-Time Automatic Drum Transcription Using Dynamic Few-Shot Learning," in Proc. *IEEE 5th Int. Symp. Internet of Sounds (IS2)*, pp. 1–8, Erlangen, Germany (2024 Sep./Oct.), https://doi.org/10.1109/IS262782.2024.10704130.

[10] A. Fernandez, "Onsets and Velocities: Affordable Real-Time Piano Transcription Using Convolutional Neural Networks," in Proc. *31st European Signal Processing Conf. (EUSIPCO)*, pp. 151–155, Helsinki, Finland (2023 Sep.).

[11] T. Collins, S. Böck, F. Krebs, and G. Widmer, "Bridging the Audio-Symbolic Gap: The Discovery of Repeated Note Content Directly From Polyphonic Music Audio," in Proc. *53rd Int. Conf. Audio Eng. Soc. Semantic Audio*, pp. 1–12, London, UK (2014 Jan.).

[12] E. Cakir, T. Heittola, H. Huttunen, and T. Virtanen, "Polyphonic Sound Event Detection Using Multi Label Deep Neural Networks," in Proc. *Int. Joint Conf. Neural Networks*, pp. 1–7, Killarney, Ireland (2015 Jul.), https://doi.org/10.1109/IJCNN.2015.7280624.

[13] A. Laaksonen and K. Lemström, "Discovering Distorted Repeating Patterns in Polyphonic Music," *J. Math. Music*, vol. 15, no. 2, pp. 99–111 (2021 Apr.), https://doi.org/10.1080/17459737.2021.1896811.

[14] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780 (1997 Nov.), https://doi.org/10.1162/neco.1997.9.8.1735.

[15] K. Cho, B. van Merriënboer, C. Gulcehre, et al., "Learning Phrase Representations Using RNN Encoder-Decoder for Statistical Machine Translation," in Proc. *2014 Conf. Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1724–1734, Doha, Qatar (2014 Oct.), https://doi.org/10.3115/v1/D14-1179.

[16] B. Janssen, W. B. de Haas, A. Volk, and P. van Kranenburg, "Finding Repeated Patterns in Music: State of Knowledge, Challenges, Perspectives," in M. Aramaki, O. Derrien, R. Kronland-Martinet, and S. Ystad (Eds.), *Sound, Music, and Motion*, pp. 277–297 (Springer, Cham, Switzerland, 2014), https://doi.org/10.1007/978-3-319-12976-1_18.

[17] O. Nieto and M. M. Farbood, "Identifying Polyphonic Musical Patterns From Audio Recordings Using Music Segmentation Techniques," in Proc. *Int. Society for Music Information Retrieval Conf. (ISMIR)*, pp. 411–416, Taipei, Taiwan (2014 Oct.).

[18] A. Laaksonen, K. Lemström, and O. Björklund, "Transposition and Time-Scaling Invariant Algorithm for Detecting Repeated Patterns in Polyphonic Music," in *Mathematics and Computation in Music*, pp. 168–179 (Springer, Cham, Switzerland, 2022), https://doi.org/10.1007/978-3-031-07015-0_14.

[19] O. Dikmen and A. Mesaros, "Sound Event Detection Using Non-Negative Dictionaries Learned from Annotated Overlapping Events," in Proc. *IEEE Workshop Applications of Signal Processing to Audio and Acoustics (WASPAA)*, pp. 1–4, New Paltz, NY, USA (2013 Oct.), https://doi.org/10.1109/WASPAA.2013.6701861.

[20] J. Forth and G. Wiggins, "An Approach for Identifying Salient Repetition in Multidimensional Representations of Polyphonic Music," *J. New Music Res.*, vol. 38, no. 3, pp. 267–292 (2009 Feb.).

[21] O. Lartillot and P. Toiviainen, "Motivic Matching Strategies for Automated Pattern Extraction," *Music. Sci.*, vol. 11, no. 1_suppl, pp. 281–314 (2007 Mar.), https://doi.org/10.1177/102986490701100111.

[22] N. Silva and L. Turchet, "Real-Time Pattern Recognition of Symbolic Monophonic Music," in Proc. *19th Int. Audio Mostly Conf. (AM)*, pp. 308–317, Milan, Italy (2024 Sep.), https://doi.org/10.1145/3678299.3678329.

[23] N. Silva and L. Turchet, "Real-Time Audio Pattern Detection for Smart Musical Instruments," *J. Audio Eng. Soc.*, vol. 74, no. 3, pp. 130–140 (2026 Mar.), https://doi.org/10.17743/jaes.2022.0250.

[24] D. Stefani, S. Peroni, and L. Turchet, "A Comparison of Deep Learning Inference Engines for Embedded Real-Time Audio Classification," in Proc. *Int. Conf. Digital Audio Effects (DAFx)*, pp. 301–307, Vienna, Austria (2022 Sep.).

[25] T. Pelinski, R. Diaz, A. L. B. Temprano, and A. McPherson, "Pipeline for Recording Datasets and Running Neural Networks on the Bela Embedded Hardware Platform," in Proc. *Int. Conf. New Interfaces for Musical Expression*, pp. 325–330, Mexico City, Mexico (2023 Jun.).

[26] L. Vignati, S. Zambon, and L. Turchet, "A Comparison of Real-Time Linux-Based Architectures for Embedded Musical Applications," *J. Audio Eng. Soc.*, vol. 70, no. 1/2, pp. 83–93 (2022 Jan./Feb.), https://doi.org/10.17743/jaes.2021.0052.

[27] A. McPherson and V. Zappi, "An Environment for Submillisecond-Latency Audio and Sensor Processing on BeagleBone Black," in Proc. *138th Conv. Audio Eng. Soc.*, paper 9331:1–7, Warsaw, Poland (2015 May).

[28] L. Turchet and C. Fischione, "Elk Audio OS: An Open Source Operating System for the Internet of Musical Things," *ACM Trans. Internet Things*, vol. 2, no. 2, pp. 1–18 (2021 Mar.), https://doi.org/10.1145/3446393.

[29] N. Silva, A. Boem, and L. Turchet, "Interactive IoMusT-Based Concerts: Real-Time Pattern Recognition and Audience Experience," in Proc. *Immersive and 3D Audio: From Architecture to Automotive (I3DA)*, pp. 1–11, Bologna, Italy (2025 Sep.), https://doi.org/10.1109/I3DA65421.2025.11202042.

[30] H. Sakoe and S. Chiba, "Dynamic Programming Algorithm Optimization for Spoken Word Recognition," *IEEE Trans. Acoust. Speech Signal Process.*, vol. 26, no. 1, pp. 43–49 (1978 Feb.), https://doi.org/10.1109/TASSP.1978.1163055.

[31] N. Silva and L. Turchet, "A Structural Similarity Index Based Method to Detect Symbolic Monophonic Patterns in Real-Time," in Proc. *25th Int. Conf. Digital Audio Effects (DAFx)*, pp. 161–168, Vienna, Austria (2022 Sep.).

[32] W. J. Youden, "Index for Rating Diagnostic Tests," *Cancer*, vol. 3, no. 1, pp. 32–35 (1950 Jan.), https://doi.org/10.1002/1097-0142(1950)3:1%3C32::AID-CNCR2820030106%3E3.0.CO;2-3.

[33] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," presented at the *3rd Int. Conf. Learning Representations (ICLR)*, pp. 1–15, San Diego, CA, USA (2015 May).

THE AUTHORS



Nishal Silva



Luca Turchet

Nishal Silva is a postdoctoral researcher in the Department of Information Engineering and Computer Science at the University of Trento. He received his Ph.D. in information engineering and computer science from the University of Trento, his M.Sc. in telecommunication and electronic engineering from Sheffield Hallam University, and his B.Eng. (Hons) in electronic engineering from Sheffield Hallam University. His research interests are in music information retrieval and smart musical instruments.

•

Luca Turchet is an associate professor in the Department of Information Engineering and Computer Science at the University of Trento. He received master's degrees in computer science from the University of Verona, in classical guitar and composition from the Music Conservatory of Verona, and in electronic music from the Royal College of Music of Stockholm. He received the Ph.D. in Media Technology from Aalborg University Copenhagen. He is a Fellow of the Audio Engineering Society.