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.

Real-Time Audio Pattern Detection for Smart Musical Instruments

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

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

This paper presents an algorithm for real-time polyphonic audio pattern detection designed to be integrated into smart musical instruments. Such instruments will enable musicians to use predefined musical patterns as triggers for external peripherals, such as stage lights, haptic wearables, and mixed reality headsets, during live performances. The paper first introduces a data set of polyphonic patterns containing musical patterns with expressive variations recorded by 20 musicians. Secondly, it presents an approach to classify audio segments in real time, consisting of a recurrent neural network architecture. Thirdly, it compares the proposed method against a traditional dynamic time warping approach as a benchmark. Results show that, although the benchmark achieves higher precision (0.8 versus 0.74), the proposed method yields a better overall performance, with an F1 score of 0.76 compared with 0.65. Most importantly, the proposed method exhibits a lower computational inference time (14 ms versus 1038 ms) and reduced CPU usage (31.36% versus 48.9%) when deployed on a Raspberry Pi 4. The low inference time meets the latency requirements necessary for real-time musical applications, and the system demonstrates the feasibility of embedded intelligence in musical instruments that can recognize polyphonic patterns with expressive variations during live performances.

Keywords: smart musical instruments, real-time pattern recognition, Internet of Musical Things

1 INTRODUCTION

The evolution of music technology has been marked by key innovations that have transformed how music is created, performed, and experienced. From the invention of the electric guitar to the development of digital audio workstations, each technological leap has opened new avenues for musical expression. Researchers and musicians have been exploring the integration of sensing and embedded computing technologies into both traditional and novel musical instruments. In more recent years, this focus has shifted toward leveraging advancements in sensors, machine learning, and connectivity to make musical instruments smarter and more interactive. Such developments have led to the emergence of a class of smart musical instruments (SMIs), which typically run real-time music information retrieval (MIR) methods on embedded devices [1]. These instruments are a fundamental component of the Internet of Musical Things [2], an extension of the Internet of Things paradigm to the musical domain.

The detection of musical patterns is a fundamental task in the field of MIR, with wide-ranging applications, including computational musicology, audio fingerprinting, music classification, and music recommendation. The presence of repeating patterns is a significant aspect of music because humans primarily recognize structure in music through the perception of repetition within a piece [3, 4]. Although despite the considerable research that has been conducted on pattern detection in recorded music, its real-time counterpart remains relatively unexplored, despite its significant potential for innovative applications such as those of SMIs.

The field of MIR has seen significant advancements in polyphonic pattern recognition [5]. Polyphony implies the existence of multiple simultaneous musical notes in music, where pattern recognition becomes increasingly complex, as opposed to its monophonic counterpart (where only a single musical note may exist at a given time). Pattern recognition in polyphony is particularly complex when attempted in real time, especially within resource-constrained embedded devices.

Real-time polyphonic pattern recognition requires algorithms capable of processing complex audio signals with minimal delay. Traditional methods may struggle with

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

real time and polyphony [6–8]. Moreover, the computational demands of real-time processing can be significant, necessitating efficient and optimized solutions for use on embedded systems to be integrated within musical instruments.

This paper presents an algorithm created to facilitate the creation of SMIs with real-time pattern detection (RTPD) capabilities. These instruments will allow the musician to use a set of predefined musical patterns as triggers for external peripherals, such as smoke machines, stage lights, haptic wearables, and mixed reality headsets. These devices can be used to supplement the music with other sensory stimuli, effectively providing musicians with an extended set of controls through phrases performed on their instruments. For instance, a potential use case would be when a musician can configure the system to recognize a specific musical phrase to trigger a stage light. The musician can play the phrase in a suitable section in the performance while the system will process the instrument output continuously to detect whether the pattern has been played. Upon the musician playing the pattern and its subsequent detection, a control message can be sent to the light controller software to trigger the stage light, effectively allowing the musician to incorporate the light into their performance.

Previous work by the authors explored the integration of symbolic monophonic pattern detection capabilities into a prototype smart electric guitar[1] [9]. The detection of patterns from a monophonic stream represents a much simpler task compared with polyphony. Additionally, the use of symbolic data, such as MIDI, make the computations much simpler but also limits the practical applicability of the method to MIDI-enabled instruments. The use of acoustic or electric instruments necessitates a perfect audio-to-MIDI conversion in real time, which despite significant recent advancements is still an unsolved problem [10].

The aim of this study is to address the limitations identified in previous works while improving the system to work with polyphony and audio. This paper highlights the algorithm development and the design of the SMIs for polyphonic audio RTPD. Moreover, the proposed method is compared against a traditional digital signal processing method as a benchmark. This research also shows that the proposed algorithm has good accuracy, latency acceptable for real-time usage, and low enough computational load to run on embedded devices.

This paper also presents a novel data set specifically designed to understand the level of expressive variation that musicians may add when repeating musician patterns in a live performance. The data set was constructed with the contribution of keyboard players and guitar players. The musicians were free to use a combination of monophony and polyphony when they were asked to first compose and record a pattern and then repeat the same pattern while incorporating expressive variations they would use in a live setting.

[1] A video of the system in operation is available at: https://youtu.be/gkOqfOT8zXM.

2 RELATED WORKS

A comprehensive review and analysis of offline pattern recognition techniques applied to polyphonic music transcription is provided in [11]. The authors discussed various approaches, including signal processing methods, machine learning algorithms, and hybrid systems. Their work provided a foundation for subsequent research in automatic music transcription. An extensive overview of the field of pattern discovery in music is reported in [12]. The authors reviewed existing techniques, discussed challenges, and proposed future research directions.

The authors of [13] presented an offline method for discovering repetitive note patterns in polyphonic music using music segmentation techniques. Their approach, based on the structural features algorithm, identifies repeated patterns by analyzing pitch and rhythm information. The method showed promising results in detecting both exact and approximate repetitions in complex polyphonic pieces, demonstrating its potential for music analysis and information retrieval tasks.

The paper reported in [14] introduced novel algorithms for matching patterns in symbolically encoded polyphonic music, allowing for pattern matching under transposition invariance. Their approach demonstrated improved accuracy and efficiency compared with previous methods, particularly in handling complex polyphonic structures.

The study described in [8] focused on identifying repetitions under transposition and time-warp invariances in polyphonic symbolic music. The researchers developed algorithms that could detect patterns even when they were distorted through transposition or temporal variations. This work significantly advanced the field of music pattern recognition by addressing the complexities of real-world musical variations.

A statistical method for complete polyphonic music transcription is proposed in [15]. The approach combined multipitch detection with rhythm quantization to estimate score times of note onsets and offsets from polyphonic MIDI performances. The method showed improved accuracy in transcribing complex polyphonic pieces.

The authors of [16] discussed various evaluation metrics for polyphonic sound event detection systems. They analyzed the strengths and weaknesses of different metrics, proposing new measures that better capture the performance of detection algorithms in realistic environments. Their work contributed to standardizing evaluation practices in the field.

The study reported in [7] applied deep neural networks to polyphonic sound event detection. A multilabel classification approach was developed using convolutional and recurrent neural networks (RNN). The method demonstrated improved performance in detecting overlapping sound events compared with traditional approaches.

An approach to detecting overlapping sound events in polyphonic audio was proposed in [17]. The authors developed a method based on non-negative matrix factorization to learn dictionaries from annotated polyphonic recordings.

Their approach showed promising results in separating and identifying concurrent sound events.

The authors of [6] addressed the challenge of discovering repeated note content directly from polyphonic music audio. They developed algorithms that could identify repetitive patterns without relying on symbolic representations, significantly advancing the field of audio-based music analysis.

A method for detecting repetitions in complex polyphonic textures across multiple musical dimensions, including pitch, rhythm, and timbre, is reported in [18]. Such a work contributed to the understanding of musical structure in polyphonic compositions.

The authors of [19] introduced a method for extracting motivic patterns in polyphonic music based on adaptive redundancy filtering. Their approach could identify significant musical motifs in complex polyphonic textures, providing valuable tools for music analysis and composition.

Taken together, the studies above show that existing research on polyphonic audio classification has explored various types of signal processing and machine learning techniques. These techniques have been used to handle the complexity of multiple simultaneous notes involved with polyphonic music. All studies have focused on offline applications (i.e., finding all occurrences of a given musical pattern in a musical score or recording). However, real-time detection remains a challenge scarcely investigated.

Real-time audio processing techniques have been developed to minimize latency and ensure seamless integration with live performances [20, 21]. These techniques often involve optimizing algorithms for speed, efficiency, and leveraging specialized hardware and embedded real-time operating systems [22, 23]. Embedded real-time audio processing is critical to the development of SMIs and is crucial for applications such as live performances, interactive installations, and real-time effects processing. However, the integration of real-time polyphonic pattern detection within an SMI remains underexplored along with the definition of methods leveraging such integration for live performance purposes. This research addresses this gap by developing a system that combines these elements.

3 DATA SET DEVELOPMENT

One of the fundamental challenges in RTPD is the variations that musicians might bring to a musical phrase when playing live. The reduced availability of data adds another layer of complexity to the challenge. Most publicly available data lack musical patterns containing expressive variations that one would encounter during a real performance. To overcome this obstacle, the Dataset of Polyphonic Patterns[2] was created. The data set consists of musical patterns and their repetitions containing expressive variations in audio. This data set is an extension to the previously published Dataset of Monophonic Patterns (DoMP)[3] [9], which contained monophonic musical patterns with expressive variations.

Similar to the methodology adopted for building the previous data set, each musician was asked to compose ten distinct musical patterns along with nine more repetitions of each pattern. Each musician was given the artistic freedom to add any variations or expressive techniques they preferred and the freedom to use a combination of both monophony and polyphony—as long as the repetition remained perceptually equivalent to the original pattern: the idea being that an untrained human listener should identify each variation to be the same as the original pattern.

The data set was made with the contribution of ten keyboard players and ten guitar players. The 20 musicians (all male, aged between 26 and 55 years, mean age $= 38.5$, SD $= 9.2$) have various musical backgrounds (classically trained musicians, contemporary studio/session musicians, funk musicians, jazz musicians, and rock musicians). The mean duration of the data set is 8.4 s ($\pm$5.54). The mean duration of the ground truth patterns is 7.4 s ($\pm$2.43).

The data set was recorded with multiple instruments because most musicians preferred to use their own instruments. Thus, the data set contains an excellent representation of different spectral and timbral characteristics of multiple instruments. The guitarists also experimented with different pickup selections, which can also be attributed as timbral variations of the pattern. For the guitar, the direct unprocessed input was obtained, and for the keyboard musicians were instructed to refrain from using tones with time-based effects such as delay or reverb.

The data set is in Waveform Audio File Format. The file naming structure is kept consistent with the previous data sets. Each file is named $artistID\_patternID\_versionID.wav$, where $\mathbb{Z}$ represents a set of integers and $artistID \in \mathbb{Z} : 60 \leq artistID \leq 79$; $patternID \in \mathbb{Z} : 0 \leq patternID \leq 9$; $versionID \in \mathbb{Z} : 0 \leq versionID \leq 9$. *artistID* represents the musician. The keyboard players are numbered 60–69, and the guitarists are numbered 70–79. *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 availability of adequate training data is a primary requirement for any deep learning approach. For the work proposed by this paper, the training data set must represent as many of the expressive variations as possible. To obtain such a training data set, the pattern variations present in the collected recordings were analyzed to identify the most common variations. Each pattern and its variations retained the same perceptual melodic structure but with elements added as extra or elements removed. Some of the expressive variations discovered by the analysis here are as follows:

- Changing the duration of notes and pauses;
- Doubling one, or a number of notes;
- Adding pitch modulations on one or several notes;
- Using the pitch-bend controller or bending the string to reach a subsequent note;

[2] The data set is available at: https://doi.org/10.5281/zenodo.14497998.

[3] https://doi.org/10.5281/zenodo.10818617

- Changing the loudness of notes;
- Adding chromatic passing notes between notes;
- Removing notes and/or pauses;
- Addition of extra notes and/or pauses;
- Alternating between staccato and legato style playing;
- Changing octaves and/or transpositions.

4 METHODOLOGY

4.1 System Architecture for Smart Musical Instruments

As mentioned earlier, the presented algorithm is designed to enable a class of SMIs with RTPD capabilities. The embedded system containing the system can be outfitted to any conventional electric instrument. A Raspberry Pi 4 is selected as the embedded system due to its availability and ease of use, and a HiFIberry DAC+ADC is used as the audio interface. The instrument output is split and sent to both the RTPD system and external amplifiers or public address systems separately. Fig. 1 shows a block diagram of the SMI design. The data flow involves capturing the audio output from the instrument, extracting features, and processing it through a virtual studio technology (VST) plug-in to detect a set of predefined patterns.

The wireless connectivity in the Raspberry Pi 4 is employed to send control messages upon successfully detecting a pattern. The control messages use the Open Sound Control (OSC) protocol. The OSC messages can be configured to trigger or control different devices that are connected to the network.

Similar to the authors' previous SMI prototype [9], the software components required to operate the SMI include ELK Audio OS and the presented algorithm housed within a VST plug-in. ELK Audio OS is an ultra-low latency operating system optimized for audio applications [24] that provides a real-time kernel and audio drivers that minimize latency. The VST plug-in hosts the algorithm and handles the real-time audio input from the instrument.

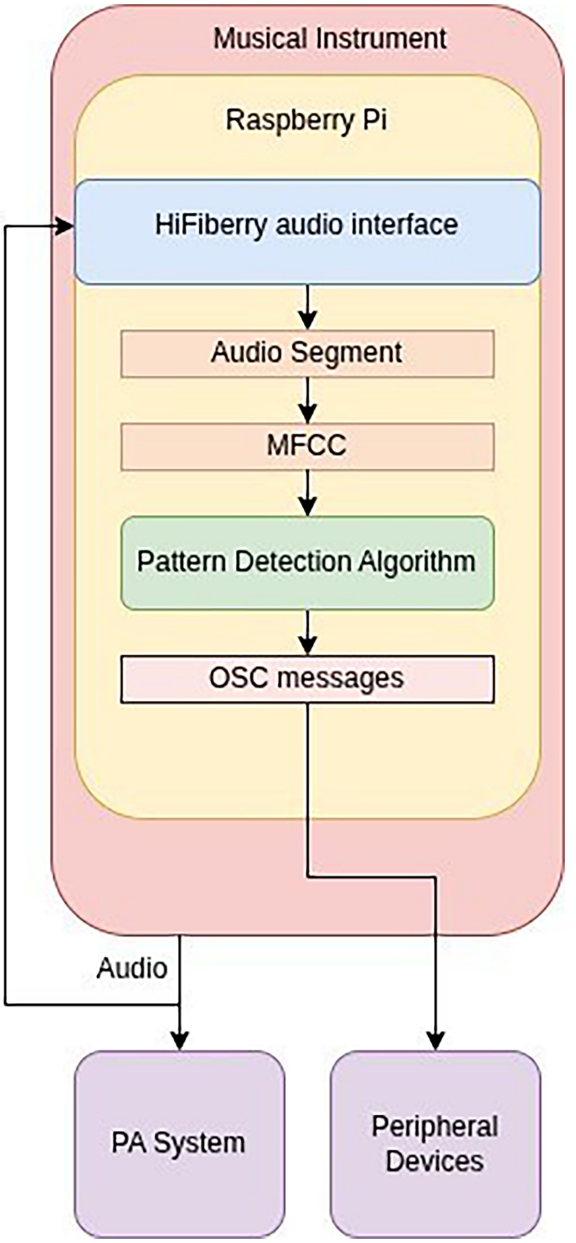

Fig. 1. A block diagram of the hardware and software setup of the system embedded in the musical instruments. PA = public address.

4.2 Polyphonic Audio Pattern Detection

The core of the pattern recognition algorithm is a trained RNN that is able to classify an audio segment from the instrument into a list of patterns previously defined. The evaluations show that the RNN is able to classify an input in real time (i.e., approximately 15 ms inference time as reported in Table 4).

The output of the musical instrument is fed directly to the algorithm (see Fig. 1) at a sample rate of 48 kHz, and a sliding audio window is extracted every 30 ms (see Fig. 2). The 30-ms interval was selected because it is approximately the temporal resolution of the human hearing system to distinguish between two sequential sound events [25]. Moreover, it ensures that the processing and inference is completed before the next window is captured. The size of the extracted audio window can be configured based on the size of the patterns that will be detected (i.e., if the patterns are

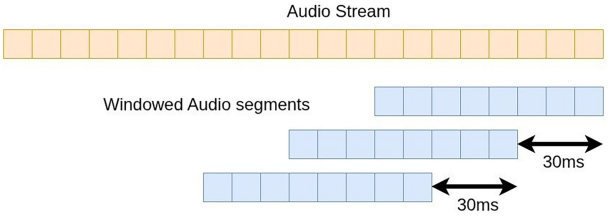

Fig. 2. Extraction of windowed segments from the incoming audio stream.

Algorithm 1. Algorithm for RNN classification.

```
Parameters:
Audio window
procedure Process Audio Stream
  while Audio input is active do
    Extract audio window every 30ms
    Compute MFCC of window
    Classify MFCC using RNN:
      pred_prob ← RNN(MFCC)
      pred_class ← arg max(pred_prob)
    if pred_class = 1 then
      Send OSC command for Pattern 1
    else if pred_class = 2 then
      Send OSC command for Pattern 2
      ⋮
    else if pred_class = n then
      Send OSC command for Pattern n
    end if
  end while
end procedure
```

approximately 3 s in length, an audio window of the same length should be used for optimal performance).

Upon extracting the window, the Mel frequency cepstral coefficients (MFCCs) are calculated to extract the audio features. The MFCCs are a suitable candidate because they are designed to mimic the human auditory system's response to sound. Moreover, the MFCCs enable a compact representation that allows for the efficient encoding of the most important aspects of the audio while simultaneously reducing the dimensionality of the data. MFCCs also capture both temporal and spectral characteristics of the audio, which is particularly important for music analysis where both rhythm and pitch information are relevant, as in the case of pattern detection tasks.

When calculating the MFCCs, a fast Fourier transform (FFT) size of 2048 and a hop size of 512 was used, resulting in overlapping frames. For a pattern of threes, this amounts to approximately 300 MFCC frames for classifying a single audio window, with each frame containing ten coefficients.

These values were experimentally identified to be the best compromise between accuracy and speed on the embedded device. The computed MFCCs of the extracted audio window is used as the input to the RNN for classification. The RNN was trained using a synthetically generated data set that consists of artificially created variations for each pattern that will be detected. Upon classification of the window, the corresponding message is transmitted. Algorithm 1 shows a pseudocode of the detection algorithm.

4.2.1 Recurrent Neural Network Model

The RNN model serves to classify each extracted windowed audio segment into a pattern. Similar to the authors' previous approach [9], a stacked RNN configuration was selected for the task due to its capability in working with synchronous and temporal data. This configuration consisted of i) a fully connected RNN layer, ii) a long short-term memory (LSTM) layer, and iii) a gated recurrent unit (GRU) layer. Fig. 3 illustrates the three types of RNN configurations used in this study. The configuration was selected empirically upon evaluating multiple configurations, including different RNN layer sizes, different FFT sizes, hop sizes etc.

Table 1. Summary of the neural network architecture.

Layer	Output shape	Parameters
Masking	(None, T, F)	0
LSTM	(None, T, 256)	1,287,168
GRU	(None, T, 128)	148,224
SimpleRNN	(None, 256)	98,560
Dense	(None, num_labels)	257 × num_labels

Note: T denotes the maximum pattern length, F the feature dimension, and num_labels the number of output classes.

Prior to training, the training data set was zero-padded to equalize the dimensions of the data. The model consisted of a masking layer to ignore the padded values in the training data set. The masking layer was followed by the stack of recurrent layers: the LSTM layer with 256 units, the GRU layer with 128 units, and the SimpleRNN layer with 256 units. The final output was produced by a dense layer with a *tanh* activation function corresponding to the number of patterns. Table 1 summarizes each layer, output shape, and number of parameters.

The algorithm operates in three distinct steps:

1. **Feature extraction**: The MFCCs are obtained for the extracted audio segment. The FFT size, hop size, and the number of Mel frequency bins are selected to facilitate a real-time performance while having the highest possible accuracy. The extracted MFCC spectrum is the input to the RNN model.
2. **Classification**: The MFCC of the window is given as input to the model. As discussed above, the RNN model is relatively light to facilitate a real-time operation while having the highest possible accuracy. The model is trained using a synthetic training data set created to mimic possible expressive variations for a list of patterns.
3. **Control message generation**: If the extracted audio segment is similar to a pattern, an OSC message corresponding to that pattern is transmitted as a trigger to external devices. The OSC message is transmitted through an HTTP port on a local network.

4.3 Synthetic Training Data Set

A primary requirement for any artificial intelligence model is an adequate training data set to represent the latent space of the data that will be classified. Because the work presented by this paper aims to develop a model suitable for live performances, the training data set must represent an adequate amount of expressive variation that a musician may apply to the pattern.

Similar to the author's previous approach [9], a rule-based model was used to create variations of each ground truth, which are musical patterns. The same rules from the authors' previous study were applied on the MFCC spec-

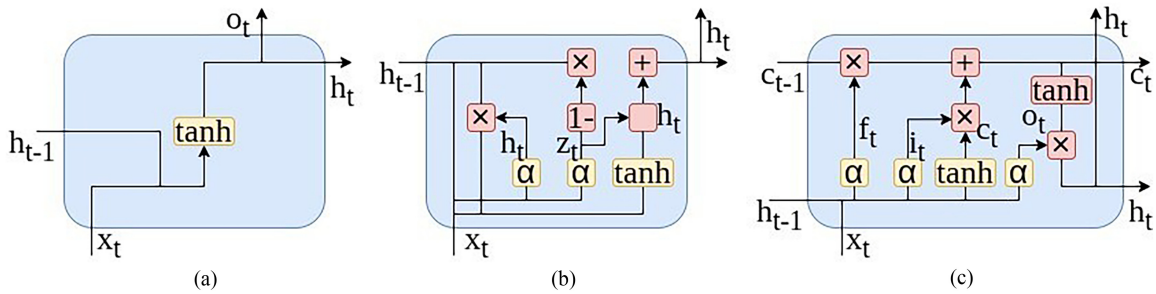

Fig. 3. A traditional RNN layer (a); a GRU layer (b) with a reset gate r_t and an update gate z_t; and a LSTM layer (c) with a forget gate f_t, input gate i_t, and output gate o_t. At each timestep t: x_t, o_t, h_t, and c_t are the input, output, hidden state, and intermediate cell state, α and *tanh* are the sigmoid and tanh activation functions, respectively.

trograms of the ground truths. The rules are applied to each Mel band within the spectrogram to create the synthetic training data set. Thus, the synthetic training data set is not actual audio variations of the ground truth but spectrograms that reflect possible variations influenced by those present in the recorded data set.

The authors' previous study identified that that approximately up to 24% of the notes of a pattern may be added notes, and approximately 3% of the notes of a pattern may be removed. An average number of 9% of the pauses in a pattern may be added, and approximately 1% of the pauses may be removed. The synthetic training data set consists entirely of MFCC spectrograms of possible variations of each ground truth. Variations, such as doubling each note, removing some notes, and changing amplitudes of notes, are represented in the training data set. Some studies discuss the application of rule-based models to mimic certain musical styles [26]. However, this study attempts to generalize said variations to suit the styles of multiple musicians. Longer patterns have more potential for variations and vice versa.

Synthetic training data are generated by applying transformations directly to the MFCC of a ground truth rather than original audio audio. Approximately 10,000 such variations per ground truth were identified to be sufficient. The rule-based variations include the following:

- **Temporal variations**: stretching/compressing time frames ($\pm 15\% - \pm 25\%$);
- **Amplitude variations**: scaling MFCC coefficients by factors of $0.8 - 1.2$;
- **Note additions**: duplicating and inserting MFCC frames at appropriate positions;
- **Note removals**: deleting MFCC frames;
- **Spectral variations**: adding Gaussian noise ($\delta = 0.05$) to MFCC coefficients.

4.4 Dynamic Time Warping

A dynamic time warping (DTW)–based approach was used as a benchmark to compare the proposed work. DTW is a time-series analysis method for temporal sequences that may vary in speed and has applications in domains such as speech recognition and stock market analysis. The similarity between two sequences is computed by constructing a distance matrix between each point and measuring the shortest path. This property of DTW makes it an ideal candidate to measure similarity between musical sequences of different lengths, such as that of a musician playing a pattern with some variation.

Moreover, DTW can handle temporal variations without relying on a training data set, and it has characteristics that provide meaningful comparison with deep learning approaches. Deterministic and probabilistic methods [27] were evaluated in the authors' previous work, along with techniques such as the structural similarity matrix [28].

For two sequences x and x' that lie in the same dimensional space with respective lengths n and m, DTW distance can be expressed as

$$\mathbf{DTW}_q(x, x') = \min_{\pi \in A(x,x')} \left(\sum_{(i,j)\in\pi} d(x_i, x'_j)^q \right)^{\frac{1}{q}}, \qquad (1)$$

where an alignment path π is a sequence of K index pairs $((i_0, j_0), (i_1, j_1), ..., (i_{K-1}, j_{K-1})$ that represent the shortest possible path between x and x' and $A(x, x')$ is the set of all possible paths. The Euclidean distance ($q = 2$) was selected as the base distance function due to its widespread use in MFCC-based audio analysis. Alternate distance functions, such as Manhattan or cosine distance, were considered, but they showed inferior performance in preliminary experiments. Hence, the distance function d can be expressed as the Euclidean distance between the two MFCC vectors: $(d(x_i, x'_j) = (||x_i - x'_j||_2)$.

The following criteria must be met for the alignment path π to be considered admissible:

- The beginning and the end of the time series must be matched: $\pi_0 = (0, 0)$, and $pi_{K-1} = (n - 1, m - 1)$;
- The sequence should increment in both i and j, and all indexes should be present: $i_{k-1} \leq i_k \leq i_{k-1} + 1$, and $j_{k-1} \leq j_k \leq j_{k-1} + 1$.

Similar to the RNN classification, two MFCC spectrograms are compared to compute the DTW between them. Because the DTW is computed between two matrices, a multidimensional DTW is required, which may be achieved by a singular warping path across all dimensions or by

merged warping paths of each individual dimension [29, 30]. The latter approach was chosen because it allows one to merge the DTWs between each pair of Mel bands. For any n ground truth patterns, n such DTW distances are computed. A static threshold is used to determine whether the audio window is similar to a pattern in the list. The threshold was calculated using a small validation set obtained randomly (20% of each musician's data). The DTW measures for each pattern and the validation set were computed, and the threshold was selected to maximize the F1 score using Youden's index. This threshold was fixed when testing, but it remained unique to each ground truth.

5 EVALUATION

The two methods were evaluated using the data set introduced in SEC. 3. The data set contains ten patterns recorded by each musician, with nine repetitions of each pattern. For the evaluation, the first pattern is considered as the ground truth against which the nine repetitions were matched. The evaluations were conducted on isolated pattern–ground truth pairs rather than on a continuous audio stream. This approach was purposefully chosen because the DTW method cannot feasibly compute pattern matches at the 30-ms intervals needed for real-time operation.

Each ground truth was matched against all the patterns recorded by the same musician. The two methods were judged on the basis that the repetitions of the ground truth should show a positive match, and any other patterns should show a negative match. The evaluation for the two methods was carried out as follows:

- **RNN method**: The synthetic data set was created, and the model was trained for each ground truth. The *actual* variations recorded by the musician and the other patterns recorded by the same musician were used in the evaluation.
- **DTW method**: The DTW was measured against the ground truth and each of its variations. To maintain a fair comparison, the DTW between the ground truth and the other patterns recorded by the same musician were also computed. A static threshold was implemented to obtain the best results.

In a real-world usage, a musician would use more than a single repeating pattern within a composition. Hence, evaluations were also done for the RNN method using groups of three patterns. The RNN model was trained using the synthetic variations of three ground truths, and the RNN would classify any input into four classes: one for each ground truth and one to denote no pattern. The evaluation of the DTW would be the same despite the number of patterns meant to be detected.

The end goal of the evaluations was to identify a suitable candidate to be deployed on embedded devices. Hence, the average memory usage and detection time for each method was computed.

Table 2. Comparison of RNN and DTW methods in detecting a single pattern.

	Precision	Recall	F1 Score
RNN	0.745 ± 0.082	0.775 ± 0.091	0.760 ± 0.087
DTW	0.801 ± 0.054	0.555 ± 0.073	0.655 ± 0.061

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

	Precision	Recall	F1 score
RNN	0.717 ± 0.075	0.693 ± 0.080	0.705 ± 0.077
DTW	0.801 ± 0.050	0.555 ± 0.065	0.655 ± 0.058

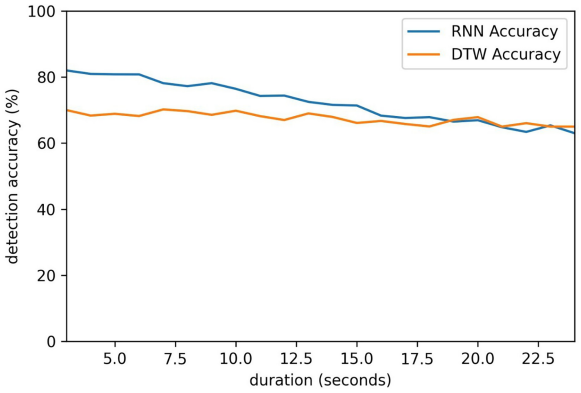

Fig. 4. Accuracy versus length.

6 RESULTS

The experiment was set up to obtain a fair comparison between the two methods. Here, the focus is on the precision, recall, and F1 score of each method to measure their accuracy, the computational load to measure their suitability to be deployed on embedded systems, and the inference time to measure the ability for real-time usage.

Table 2 shows the average the precision, recall, and F1 score of each method when detecting a single pattern, and Table 3 shows the average precision, recall, and F1 score in detecting three patterns. Note that the accuracy metrics are identical for the DTW method for both evaluations because a one-to-one comparison is made for a list of ground truths regardless of the number of patterns to be detected.

Another crucial factor in accuracy is the length of the patterns. It was identified that the accuracy was proportional to the length, where longer patterns showed a decreased accuracy for the RNN model. The DTW method, on the other hand, showed no significant changes in accuracy despite the length. Fig. 4 shows the average accuracy as a function of pattern length.

Latency is a crucial aspect in any real-time system. The pattern must be identified as soon as the musician plays the last note, and the control signal must be transmitted instantaneously. The algorithm must show a low enough inference time to be suitable for real-time use. To measure the inference time and computational cost within an em-

Table 4. The average time for the pattern detection along with the average CPU and RAM usage.

	RNN	DTW
Detection time (ms)	14.69	1038.34
Average CPU usage (%)	31.36	48.9
Average RAM usage (MB)	442.6	342.3

bedded system, both algorithms were implemented within a Raspberry Pi 4.

Due to the high inference time shown by the benchmark and the large number of detections required for performance comparison, only the classification time and computational load were measured on the embedded device. Elk Audio OS possesses a significant advantage in its ultra-low latency audio processing ($\leq$1 ms) [24], thus contributing only a marginal increase to the classification latency for pattern detection.

Hence, the inference time is sufficient to validate the low-enough inference and latency values for real-time usage. Table 4 shows the average inference time, CPU usage, and RAM usage by the two methods on a Raspberry Pi 4. It can be assumed that the accuracy observed in the evaluation is representative of the embedded deployment, barring any changes in model architecture.

7 DISCUSSION

The results of the evaluations show that both approaches can be used to detect the presence of polyphonic patterns in audio. However, the different performance characteristics makes them suitable for different use cases, the RNN-based method being the obvious choice for the proposed application in creating SMIs with embedded intelligence. When comparing the results, the higher precision shown by the DTW method (0.801 versus 0.717) can be observed along with a higher recall shown by the RNN (0.693 versus 0.555), resulting in an overall higher F1 score (0.705 versus 0.656).

The higher precision of the DTW method showed a detection of fewer false positives. This is expected, given that DTW directly compares the input against ground truths, providing a one-to-one match in all use cases. The RNN, which is trained on a data set of synthetically generated variations of each ground truth, shows greater flexibility in detecting patterns with expressive variations, as evidenced by its lower rate of false positives and higher recall.

Perhaps the most significant finding of the authors' evaluations is the difference in detection time when deployed on an embedded system. The average inference latency for RNN classification was 14.69 ms, well below the threshold required for real-time audio applications. Moreover, each independent process (extracting the window/feature extraction and classification) happens within individual processing threads. Hence, the pattern detection within an extracted audio segment can be performed well in time before the subsequent audio segment is extracted in 30 ms, allowing the real-time operation.

The DTW method showed an approximate tenfold increase in detection time, which is due to the distance matrix computed by the DTW, that causes the complexity to increase quadratically with sequence length. The RNN, once trained, can process inputs with relatively consistent time regardless of pattern complexity, which makes it more usable in real-time applications that require immediate feedback.

The computational resource requirement also favored the RNN approach in terms of CPU usage (31.36% versus 48.9% for DTW). This suggests that the RNN approach is more efficient for real-time processing on embedded devices used in the SMIs in this study. This efficiency is crucial for maintaining consistent performance during extended live performances without system degradation or overheating.

Although the RNN approach shows advantages in CPU usage and latency, DTW showed superior memory efficiency (342.3 MB versus 442.6 MB). This trade-off reflects the need to load model weights and maintain activation states for the RNN, and the DTW operates with minimal memory overhead. This factor may influence implementation on devices with extreme memory constraints, where the RNN model may need further optimizations.

An interesting observation from this study's results is the relationship between pattern length and recognition accuracy (see Fig. 4). Although the DTW method maintained relatively consistent performance regardless of pattern length, the RNN model showed decreasing accuracy as pattern length increased. This suggests that longer patterns introduce more opportunities for variations that challenge the predictive capacity of the RNN. This limitation could potentially be addressed through more comprehensive training with additional variations of longer patterns or by implementing a segmentation approach that breaks longer patterns into more manageable subsequences.

The performance metrics achieved in this study (F1 scores of 0.705 for RNN and 0.656 for DTW) indicate that, although the system performs well, there remains room for improvement. It must be noted that these scores reflect performance in detecting patterns with expressive variations in a real-world context, which is inherently more challenging than controlled laboratory conditions.

Moreover, the RNN is scalable because multiple patterns can be detected using a single model. Groups of three patterns were evaluated, classifying inputs into four classes (one for each pattern plus a "no pattern" class). This shows an advantage in real-world implementations, where multiple patterns may be recognized within a single composition. The DTW, on the other hand, would require multiple separate comparisons to achieve similar functionality, potentially increasing computational overhead in multipattern scenarios.

One limitation of the proposed method is that, although the synthetic training data generation approach proved effective, it may not capture all possible expressive variations that musicians might employ. Future research could explore more sophisticated methods for generating training

data or potentially implement online learning approaches that adapt to individual musicians' playing styles over time.

It should also be noted that the evaluations consisted of ground truths and repetitions recorded by the same musician. A further development of this system would be the integration of cross-musician and cross-instrument evaluations.

Another area for future exploration is the integration of multimodal inputs. Although the current system focuses on audio input, combining this with other sensor data from SMIs could enable more complex interaction possibilities. This multimodal approach would align with the broader vision of the Internet of Musical Things by creating better interactions between performers, instruments, and environments.

8 CONCLUSION

This paper presented a real-time polyphonic audio pattern detection system that could be embedded in SMIs, enabling musicians to trigger external peripherals, such as stage lights, smoke machines, haptic wearables, and mixed reality headsets, using musical patterns played on their instruments. Two distinct approaches, an RNN-based method and a DTW-based benchmark, were evaluated, and their performance was assessed in terms of accuracy, latency, and computational efficiency.

The RNN-based approach demonstrated superior performance for real-time applications, achieving an F1 score of 0.705 with an average detection latency of 14.69 ms. The evaluations showed that the RNN method is suitable to be deployed on embedded systems for real-time operation. In contrast, whereas the DTW method showed slightly higher precision, its significantly higher latency (1038.34 ms) makes it unsuitable for real-time applications.

To support the development and evaluation of the proposed system, the Dataset of Polyphonic Patterns was created. Such a data set contains musical patterns with artistic and expressive variations. The data set was made through the contribution of 20 musicians (ten guitar players and ten keyboard players) across different musical backgrounds. This data set, along with the authors' methodology for synthetic training data generation, is a contribution to the field of MIR and SMIs.

To continue the work presented by this paper, the introduced SMIs will be prototyped, and their performance, accuracy, and usability in real-world performance contexts will be evaluated. Moreover, comprehensive evaluations of the SMIs will be conducted thorough user studies and performances involving musicians, composers, and audience members.

Modern smartphones are becoming increasingly powerful and possess dedicated graphic processing units and neural processing units that can provide a mobile platform for hosting larger neural networks with more sophisticated classifiers. As they are self-powered with built-in wireless connectivity and touch screens, they could potentially be an interesting platform to deploy such systems to convert traditional musical instruments to SMIs easily. However, mobile phones may not have the ultra-low audio latency offered by embedded computing devices. Although being attentive to audio latency, development of such systems on mobile platforms is an interesting development of the authors' current work.

Future work could explore further improvements in pattern recognition accuracy, especially for longer patterns, and more sophisticated approaches to training data generation. Additionally, on-the-fly pattern learning and adaptation to individual playing styles are promising directions for making the system more flexible. Integration with other emerging technologies, such as augmented reality environments, could further extend the possibilities for interactive musical performances.

In conclusion, this study showed the feasibility and creative potential of embedding real-time polyphonic audio pattern detection into SMIs. Such innovative applications can open new possibilities for musical expression and performance in the digital age while maintaining the critical low-latency requirements necessary for real-time musical applications.

9 ACKNOWLEDGMENT

We acknowledge the support of the Musical Metaverse (MUSMET) project funded by the European Innovation Council (EIC) Pathfinder Open scheme of the European Union (Grant Agreement 101184379). Views and opinions expressed are, however, those of the author(s) only and do not necessarily reflect those of the European Union or the European Innovation Council. Neither the European Union nor the European Innovation Council can be held responsible for them.

10 REFERENCES

[1] 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.

[2] 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.

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

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

[5] A. Laaksonen and K. Lemström, "Advanced Polyphonic Music Pattern Matching Algorithms with Timing Invariances," in Proc. *9th Intl. Conf. of Mathematics and Computation in Music (MCM)*, pp. 242–254 (Coimbra, Portugal) (2024 Jun.). https://doi.org/10.1007/978-3-031-60638-0_19.

[6] 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," presented at the *53rd Conv. of the Audio Engineering Society* (2014 Jan.), paper 1-2.

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

[8] 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.

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

[10] E. Benetos, S. Dixon, Z. Duan, and S. Ewert, "Automatic Music Transcription: An Overview," *IEEE Signal Process. Mag.*, vol. 36, no. 1, pp. 20–30 (2018 Dec.). https://doi.org/10.1109/MSP.2018.2869928.

[11] A. Pertusa, and J. M. Iñesta, "Pattern Recognition Algorithms for Polyphonic Music Transcription," in Proc. *4th Intl. Workshop on Pattern Recognition in Information Systems*, pp. 80–89 (Porto, Portugal) (2000 Apr.). https://doi.org/10.48550/arXiv.2406.15249.

[12] B. Janssen, W. B. de Haas, A. Volk, and P. van Kranenburg, "Finding Repeated Patterns in Music: State of Knowledge, Challenges, Perspectives," in Proc. *Sound, Music, and Motion: 10th Intl. Symposium, (CMMR)*, vol. 8905, pp. 277–297 (Marseille, France) (2014 Oct.). https://doi.org/10.1007/978-3-319-12976-1_18.

[13] O. Nieto and M. M. Farbood, "Identifying Polyphonic Musical Patterns From Audio Recordings Using Music Segmentation Techniques," in Proc. *15th Intl. Symposium for Music Information Retrieval Conf. (ISMIR)*, pp. 411–416 (Taipei, Taiwan) (2014 Oct.).

[14] A. Laaksonen, K. Lemström, and O. Björklund, "Transposition and Time-Scaling Invariant Algorithm for Detecting Repeated Patterns in Polyphonic Music," in Proc. *Intl. Conf. on Mathematics and Computation in Music (MCM)*, pp. 168–179 (Atlanta, GA) (2022 Jun.). https://doi.org/10.1007/978-3-031-07015-0_14.

[15] E. Nakamura, E. Benetos, K. Yoshii, and S. Dixon, "Towards Complete Polyphonic Music Transcription: Integrating Multi-Pitch Detection and Rhythm Quantization," in Proc. *IEEE Intl. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 101–105 (Calgary, Canada) (2018 Apr.). https://doi.org/10.1109/ICASSP.2018.8461914.

[16] A. Mesaros, T. Heittola, and T. Virtanen, "Metrics for Polyphonic Sound Event Detection," *Appl. Sci.*, vol. 6, no. 6, p. 162 (2016 Jun.). https://doi.org/10.3390/APP6060162.

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

[18] J. Forth and G. Wiggins, "An Approach for Identifying Salient Repetition in Multidimensional Representations of Polyphonic Music," https://eprints.goldsmiths.ac.uk/id/eprint/8578/1/forth-wiggins-2009.pdf (2009 Feb.).

[19] O. Lartillot and P. Toiviainen, "Motivic Matching Strategies for Automated Pattern Extraction," *Music. Sci.*, vol. 11, no. 1, pp. 281–314 (2007 Mar.).

[20] D. Stefani, S. Peroni, and L. Turchet, "A Comparison of Deep Learning Inference Engines for Embedded Real-Time Audio Classification," in Proc. *25th Intl. Conf. on Digital Audio Effects (DAFx)*, vol. 3, pp. 256–263 (Vienna, Austria) (2022 Sep.).

[21] 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. *Intl. Conf. on New Interfaces for Musical Expression (NIME)*, pp. 325–330 (Mexico City, Mexico) (2023 Jun.). https://doi.org/10.48550/arXiv.2306.11389.

[22] 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.). https://doi.org/10.17743/jaes.2021.0052.

[23] A. McPherson and V. Zappi, "An Environment for Submillisecond-Latency Audio and Sensor Processing on BeagleBone Black," presented at the *138th Conv. of the Audio Engineering Society* (2015 May), paper 9331.

[24] 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, p. 12 (2021 Mar.). https://doi.org/10.1145/3446393.

[25] B. Moore, *An Introduction to the Psychology of Hearing* (Brill, Boston, MA, 2013), 6th ed.

[26] M. Amerotti, S. Benford, B. Sturm, and C. Vear, "A Live Performance Rule System Informed by Irish Traditional Dance Music," in Proc. *16th Intl. Symposium on Computer Music Multidisciplinary Research (CMMR)*, pp. 127–139 (Tokyo, Japan) (2023 Nov.).

[27] N. Silva, C. Fischione, and L. Turchet, "Towards Real-Time Detection of Symbolic Musical Patterns: Probabilistic vs. Deterministic Methods," in Proc. *27th Conf. of Open Innovations Association (FRUCT)*, pp. 238–246 (Trento, Italy) (2020 Sep.). https://doi.org/10.23919/FRUCT49677.2020.9211010.

[28] N. Silva and L. Turchet, "A Structural Similarity Index based Method to Detect Symbolic Monophonic Patterns in Real-Time," in Proc. *25th Intl. Conf. on Digital Audio Effects (DAFx)*, vol. 3, pp. 161–168 (Vienna, Austria) (2022 Sep.).

[29] M. Wöllmer, M. Al-Hames, F. Eyben, B. Schuller, and G. Rigoll, "A Multidimensional Dynamic Time Warping Algorithm for Efficient Multimodal Fusion of Asynchronous Data Streams," *Neurocomputing*, vol. 73, no. 1, pp. 366–380 (2009 Dec.).

[30] M. Shokoohi-Yekta, B. Hu, H. Jin, J. Wang, and E. Keogh, "Generalizing DTW to the Multi-Dimensional Case Requires an Adaptive Approach," *Data Min. Knowl. Discov.*, vol. 31, pp. 1–31 (2017 Feb.). https://doi.org/10.1007/s10618-016-0455-0.

THE AUTHORS

Nishal Silva

Luca Turchet

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

•

Luca Turchet is an associate professor in the Department of Information Engineering and Computer Science of University of Trento. He received master degrees (summa cum laude) in computer science from University of Verona in classical guitar, composition from the Music Conservatory of Verona, and electronic music from the Royal College of Music of Stockholm. He received the Ph.D. in Media Technology from Aalborg University Copenhagen. His scientific, artistic, and entrepreneurial research has been supported by numerous grants from different funding agencies, including the European Commission, the European Institute of Innovation and Technology, the European Space Agency, the Italian Minister of Foreign Affairs, and the Danish Research Council. He is cofounder of the music-technolgy company Elk Audio. He serves as an associate editor for *IEEE Transactions on Human-Machine Systems* and has served as an associate editor for *IEEE Access* and the *Journal of the Audio Engineering Society*.