Real-Time Pattern Recognition of Symbolic Monophonic Music

Nishal Silva
Department of Information Engineering and Computer Science, University of Trento
Trento, Italy
nishal.silva@unitn.it

Luca Turchet
Department of Information Engineering and Computer Science, University of Trento
Trento, Italy
luca.turchet@unitn.it

Abstract

This paper investigates the real-time detection of predefined monophonic patterns from the MIDI output of a digital musical instrument. This enables the development of instruments and systems for live music, which can recognize when a musician plays a certain phrase and repurpose such information to trigger external peripherals connected to the instrument. Specifically, we compare the recognition performance of Dynamic Time Warping and Recurrent Neural Network-based approaches. We employ different representation formats of musical data to optimize the efficiency of each computational method. To evaluate the algorithms, a novel dataset is introduced which includes recordings from 20 keyboard players and 20 guitar players. The evaluation focuses on the algorithms' ability to recognize patterns amid variations that impede a straightforward one-to-one comparison. The results reveal that both methods perform well in detecting up to 3 distinct patterns. However, as the number of different patterns increases up to 10, dynamic time warping exhibits a negative correlation with the recognition performance, while the recurrent neural network maintains high detection accuracy of approximately 98%. Taken together, our findings demonstrate the potential of machine learning in handling complex musical patterns in real-time, paving the way for novel applications involving smart musical instruments.

CCS Concepts

• **Applied computing → Sound and music computing**; • **Computer systems organization →** *Embedded and cyber-physical systems*; • **Information systems → Music retrieval**.

Keywords

Smart Musical Instruments, Real-Time Pattern Recognition, Internet of Musical Things

ACM Reference Format:
Nishal Silva and Luca Turchet. 2024. Real-Time Pattern Recognition of Symbolic Monophonic Music. In *Audio Mostly 2024 - Explorations in Sonic Cultures (AM '24), September 18–20, 2024, Milan, Italy*. ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3678299.3678329

AM '24, September 18–20, 2024, Milan, Italy
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0968-5/24/09
https://doi.org/10.1145/3678299.3678329

1 INTRODUCTION

The presence of repeating patterns is a fundamental and important characteristic of music. Human listeners recognize structure in music through the perception of repetition and other relationships within a composition [9, 24, 33]. The field of Music Information Retrieval (MIR) has explored the computational detection of musical patterns extensively, leading to various applications such as audio fingerprinting, indexing, plagiarism detection, music classification, and recommendation systems. Various authors have proposed algorithms capable of taking a piece of music as input and producing a list, visualization, or summary of repeated patterns as output [7, 8, 13, 14, 17, 20]. However, limited attention has been given to real-time implementations: where analysis must be done as soon as the musical pattern is produced, typically by a single musical instrument. This case is particularly challenging as it is not possible to rely on pre-processing or post-processing techniques typically employed in offline contexts.

Real-time pattern detection holds significant importance in the realm of Internet of Musical Things (IoMusT) applications, which fall at the confluence of the Internet of Things and music technology domains [31]. This is particularly true for applications centered around the so-called smart musical instruments [30]. These digital musical instruments possess wireless connectivity capabilities facilitated by embedded computing systems specifically designed for real-time audio processing and networking tasks. However, to date, little research has been conducted on such scenarios envisioned in the context of the IoMusT, primarily due to the limited availability of appropriate real-time tools that are computationally efficient enough to operate on embedded systems with constrained capabilities. These constraints include limited computing power, memory, I/O ports, and the need for minimum power consumption.

The goal of the present study is to develop a real-time monophonic pattern detection system to facilitate the development of a smart musical instrument capable of identifying the occurrence of musical patterns from its output in real-time and repurpose that information as controls for connected peripheral devices (such as smoke machines, stage lights, visuals on screens, as well as smartphones of audience members in participatory live music contexts). The core of the smart musical instrument will be an embedded computing device running the pattern detection algorithm reported hereinafter, with wireless connectivity to transmit control messages.

Most musicians have a *library of licks* from which they borrow during a solo or improvisation. With access to such a smart musical instrument, a musician can map each musical pattern (*(or lick)*), to a digital control message (e.g., Open Sound Control, MIDI). The pattern detection algorithm is able to transmit the control message associated to each pattern upon detection. Thus, the musician could control external devices using the music played on their instrument. With the work presented by this paper and its subsequent developments, we aim to provide musicians with a creative tool targeting live performance, which supplements their sonic output.

One of the primary challenges we faced was establishing a definition for a musical pattern. It is highly unlikely that a musician will play the **exact** sequence of notes each time. There will be subtle variations in time and pitch due to the human nature, and there will also be variations that the musician does on purpose to add some expressiveness to the performance. The developed algorithm should be able to identify musical patterns despite any expressive variations. For this purpose, as well as to evaluate the proposed detection methods, we created a novel dataset of 4000 patterns and related expressive variations recorded from musicians. The motivation behind this stems from the lack of human-made datasets with pattern repetitions and expressive variations. The dataset consists of recordings made by 20 keyboard players and 20 electric guitar players, as these are arguably the two most widespread musical instruments used today.

In this paper, we present a comparison between a Dynamic Time Warping (DTW), and a Recurrent Neural Network (RNN) based approach. The incoming symbolic music is represented using a matrix-based method, as well as a sampled discrete time series. The different representation formats of the musical data has been selected to exploit the maximum efficiency of each computational method.

The DTW method is a one to one similarity check against a ground-truth pattern. However, the RNN method requires the adequate availability of training data, which in our case is expressive variations of each musical pattern. As it is impractical for a musician to play many repetitions of each pattern for training, we present a rule-based model to generate melodic and expressive variations of a given ground-truth pattern. The rules were set after careful study of the variations present in the dataset, and the model is able to create a synthetic training set which mimic the real-world expressive variations of each musical pattern.

In summary, the contributions of this study are:

- A comparison between DTW and RNN based algorithms to detect the presence of previously defined musical patterns from an incoming stream in real-time;
- A novel, comprehensive dataset of musical patterns along with repetitions containing expressive variations;
- A rule-based computational model to generate expressive and melodic variations of a given ground truth, to construct a synthetic training dataset for machine learning approaches.

2 Related Works

Various studies have investigated the detection of repeated patterns in music. However, most of these studies are aimed towards the detection of unknown patterns in recorded music. In recent years, the MIREX[1] task *"Discovery of repeated themes and sections"* has served as a platform for researchers to submit their work on pattern detection. There are multiple methods presented which primarily cater to the discovery of previously-unknown patterns and establishing a hierarchy of their significance.

[1]https://www.music-ir.org/mirex

The study reported in [6] presents a method able to find both monophonic and polyphonic patterns along with repeated sections in symbolic representations of music through a geometric approach. Such a method inspects all displacements between note pairs, i.e., if there is a pattern occurring twice in the piece, A and B - the distances from all notes in A to their counterparts in B should be the same.

The authors of the study reported in [19] detail a point-set comparison algorithm where the music is presented in the form of a multi dimensional point-set. The experiments deal with the polyphonic version of the JKUPDD dataset. The authors introduce the maximal translatable patterns vector - which is the set of points in the dataset that can be translated by a vector to give other points. The method presented is a greedy compression algorithm that is able to find the best maximal translatable patterns, append them to a new vector, and remove these points from the dataset.

The method presented in [22] identifies the repetitive musical patterns of a given music piece through a self similarity matrix and is able to operate on audio or symbolic representations. The signal is windowed and a chromogram is obtained for each window, and the key-transposition invariant self-similarity matrix is computed. A scoring and threshold is performed followed by a grouping by occurrence.

The algorithm described in [16] finds repetitions of sequential patterns from monophonic sequences represented in a symbolic format. The algorithm progressively analyzes the musical sequence through one single pass. The analysis is carried out for each successive note of the sequence. For each new note to the algorithm, it is checked 1) if pattern occurrence(s) ending at the previous note can be extended with the new note; 2) if the new note initiates the start of a new pattern occurrence.

The system proposed in [23] is intended for pattern discovery in symbolic representations of monophonic music and it introduces the compositional hierarchical model to provide a hierarchical representation of the audio signal. The input to the system is a symbolic representation of the music signal, consisting of a set of pitches, each defined by an onset and an offset. The model first produces compositions of two pitches, then three, and so forth. A statistical approach is employed to retain the compositions which cover the most information in the input layer. Based on the composition's occurrence, the learning process retains the compositions which are more frequently activated.

The work reported in [33] explores the different features humans would use to categorize songs into groups. The authors have identified that the contour, rhythm, and motifs of a melody are important in establishing a similarity between melodies. In particular, the re-occurrence of short characteristic motifs is an important factor in establishing the similarity of songs that belong to the same tune family. The authors present an annotated dataset of 260 folk songs categorized into 26 tune families. In computational approaches to the study of variation among folk song melodies from oral culture, both global and local features of melodies have been used. From a computational point of view, the representation of a melody as a vector of global feature values, each summarizing an aspect of the entire melody, is attractive. However, from an annotation study on perceived melodic similarity and human categorization in music presented by the same work [33], it followed that local features of melodies are most important to classify and recognize melodies.

The study presented by [32] compares between the global and local features in the classification of folk songs. The study, through an annotation study, investigates the importance of global features which summarize a melody as a vector from a computational point of view, and the importance of local features in the perception of similarity and human categorization. The categorization uses a nearest neighbor classification as well as a Euclidean distance. The study concludes that local features carry more melodic information than global features, and hence locality is a crucial factor in establishing similarity among folk song melodies.

The study conducted by [11] proposes a generalized skipgram model which allows arbitrary cost functions, filtering, recursive and memory efficient applications. As stated in the study, it is possible to employ skipgrams for the discovery of repeated patterns of close, non-simultaneous events or notes due to the the generalizations and optimizations made. The algorithm operates on symbolic representations of polyphonic music, and has been evaluated on a dataset containing several piano sonatas of Wolfgang Amadeus Mozart in MIDI format [11].

A method to detect repeating patterns in audio versions of Indian music is presented in [29], which utilizes an entire structure, as opposed to a note based repetitions. Features such as mel-frequency cepstral coefficients, modulation spectral features, and jitter are calculated to reduce the computational time observed in signal level comparison. The audio is split into frames, and a dynamic time warping is used to measure the similarity between frames. The study states that songs with high complexity contain longer patterns that repeat less often, and songs with low complexity contain shorter patterns that repeat more often.

A method to automatically detect repeating patterns in polyphonic music in the JKUPDD database, in audio or symbolic representations is presented by [34], using a variable Markov oracle. The study uses chroma features which are processed based on musical heuristics such as modulation, beat-aggregation, etc, and fed into the variable Markov oracle based framework.

The study reported in [15] presents a method based on large pretrained audio neural networks. The neural networks are trained on the large-scale AudioSet dataset, and is capable of being transferred to other tasks. The authors have transferred the pretrained audio neural networks to several pattern recognition tasks using several different datasets [15].

The authors of [10] present a method that models human judgement of what constitutes a significant pattern by incorporating annotations of repeated patterns, avoiding the need to design heuristics. The system works on symbolic representations of monophonic music, and has been evaluated on the Meertens Tune Collection Annotated Corpus dataset [10]. The algorithm uses a neural network trained to detect a low dimensional embedding of the feature space, that maps passages of music close together when they are occurrences of the same ground-truth pattern, based on human annotations.

The work described in [24] presents a hypothesis that fusing the output from various musical pattern discovery algorithms will improve the pattern discovery results. The study presents a method to combine the output from ten state-of-the-art algorithms using the MIREX dataset, and the annotated corpus of the Dutch song database [24], which are human annotated, and presents a meta-analysis of the (dis)similarities among pattern discovery algorithms' output.

The authors of [5] propose a correlation-based method to determine the similarity between songs. A statistical operation, named the correlation coefficient is introduced and evaluated on a datset of Indian classical music-based songs. The computation is based on the occurrence of 16 fundamental frequencies between two songs, and is aimed toward the development of a music recommendation system for music therapy applications.

Methods of unknown pattern discovery are usually designed to determine if repeating patterns are present in entire compositions, and require prior knowledge of the composition in its entirety to operate efficiently. However, as we are implementing a system to be used in a live environment with no knowledge of the incoming notes beforehand (but with the knowledge of which patterns to detect), solutions presented in existing literature cannot be used to accomplish our desired task.

In our earlier study [27], we presented a comparison between a deterministic method and several machine learning methods designed to detect patterns in symbolic real-time music. The deterministic method was a straight forward boundary-checking system of pitch, amplitude, and time. We presented a single neural network-based method, a multiple neural network-based method: using a dedicated neural network for each pattern, a recurrent neural network-based method, and a convolutional neural network-based method. Of all methods presented, the deterministic method had the best results, identifying 100% of the patterns with zero false detections from a synthetically created dataset. The recurrent neural network-based method had a very high number of true positives, but the number of false positives also was significant.

However, some significant limitations of this study were the fixed window size (i.e., each pattern repetition must be identical in length to the ground truth for successful detection) and the lack of *realistic* variations in the dataset (i.e., repetitions with added notes, removed notes, or pitch modulations). The motivation behind the present study was to overcome these limitations of our previous work reported in [27]. One of the most significant improvements is the reduction of false detections encountered by neural network-based methods. Another limitation we wish to overcome is the previous systems' inability to detect transposed versions of patterns. While doing so, we wish to improve the set-up time when compared with machine learning-based methods, as well as a lower computational load.

3 Dataset

The reduced availability of data for evaluation is a challenge faced by many pattern detection algorithms. This issue is exacerbated when the data lack both musical patterns and repetitions with expressive variations. As most existing datasets are constructed synthetically or extracted from studio recordings, they often fail to capture the subtleties of a live musician's performance, which is essential to our purpose of creating a musical instrument with intelligent features. To bridge this gap, we created the "Dataset of

Musical Patterns" (DoMP), a dataset of musical patterns and expressive variations resulting from the contributions of 40 musicians (20 keyboard players and 20 guitar players of various musical styles and backgrounds)[2].

The dataset contains musical patterns, along with repetitions with artistic and melodic variations in both MIDI and audio format. The dataset was created with contributions from both keyboard and guitar players. Each musician was asked to compose 10 distinct monophonic musical patterns or phrases and then repeat each 9 more times. Each musician was given the artistic freedom to add any variations or expressive techniques they prefer on each subsequent version so long as the original pattern remained perceptually equivalent: the idea being that an untrained human listener should identify each variation to be the same as the original pattern. Fig. 2 shows such an example of a pattern and a repetition with variation. Thus, the dataset contains 10 versions of each pattern: the first being the original or the ground-truth, and the next 9 versions being repetitions containing expressive and melodic variations. The dataset comprises of 4000 patterns and variations, of which 2000 are played on a keyboard and 2000 are played on an electric guitar outfitted with a MIDI tracking device.

We believe that out dataset is unique in its focus on musical patterns with artistic variations. Each pattern is accompanied by repetitions, some of which contain artistic variations. This reflects the use case of a musician repeating the same pattern within a single, or multiple performances. Moreover, the dataset was created with live recordings from musicians, and it is specifically targeted towards real-time pattern detection.

3.1 Patterns recorded with keyboard

All keyboard players preferred to use their own instrument for the recordings. The keyboard players were from different musical backgrounds, some examples being classically trained pianists, church organists, contemporary studio/ session musicians, funk musicians, solo performers, and keyboardists in rock and pop bands.

The keyboard players used different expressive techniques in their playing. We noticed that some musicians prefer certain techniques over others. Most keyboard players involved in live bands used the pitch-bend controller, while its presence was very rare among piano players and church organists. Some musicians prefer hard keystrokes, while some prefer to play very soft. Adding passing chromatic notes in sequences, using the pitch-bend to reach subsequent notes, and slight pitch modulations on notes are some examples of the expressive techniques present in the dataset.

3.2 Patterns recorded with electric guitar

The recording process for the eclectic guitar segment was somewhat restricted due to the need of a MIDI tracker[3] for the electric guitar. The MIDI tracker converts the notes played on the guitar into symbolic representation, thereby enabling them to be used with our algorithm (see Fig. 1).

Similarly to the keyboard players, the guitarists also belonged to different musical backgrounds such as classically trained guitarists, members of rock bands, blues guitarists, and jazz guitarists. Guitar players used various expressive techniques that are present in the dataset. The most common variation is deviations in time. String bending was also a very common technique used by many guitar players: either to reach the next note or to add slight vibrato effects. Hammer-ons and pull-offs were also some common techniques used.

However, several restrictions had to be applied to overcome the limitations of the MIDI tracker. Techniques such as very fast playing, fast legato sequences that span across multiple strings, and fast multi-finger tapping were often tracked as single notes with large and frequent pitch modulations. Techniques such as muted notes, ghost notes, and harmonics (artificial/ natural/ pinch) were tracked as false notes. Moreover, the MIDI tracker works on the MIDI 1.0 standard, and it lacks polyphonic pitch bending, which is a technique most guitar players use frequently, which was not a concern for us as the dataset is monophonic. The guitarists were asked to refrain from using techniques listed above to overcome the hardware limitations. Despite the restrictions imposed, the dataset contains a large number of artistic and expressive variations.

The expressive variations performed on the patterns in the dataset range from very subtle to noticeably obvious, such that perception of the original pattern is retained. However, these levels of variation may be deemed too high for a live performance, where performers are likely to bring about minimal variations to a predefined pattern. As a result, we can consider our dataset as a *worst case scenario* it represents the most challenging situations that a pattern detection algorithm may encounter. These characteristics, the inclusion of different musical instruments, along with the involvement of diverse musicians in recording the dataset make it an ideal candidate for evaluating our algorithms, as well as a benchmark for future studies.

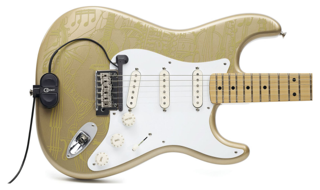


Figure 1: Electric guitar outfitted with the MIDI tracker.

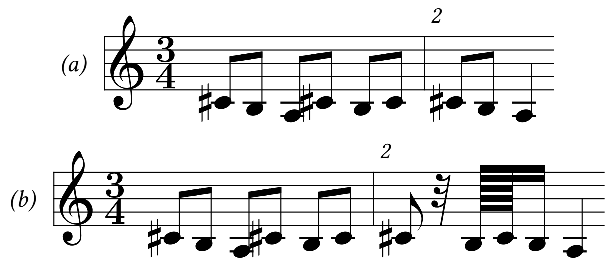


Figure 2: A pattern from the dataset *(a)*, and a version with expressive variations by including several extra notes *(b)*.

[2]The dataset is available at: https://doi.org/10.5281/zenodo.10818617
[3]https://www.fishman.com/tripleplay/

4 What is a Musical Pattern?

As elaborated in Section 3, recordings were obtained from many musicians to objectively define a suitable definition for a musical pattern. In the literature, the commonly used definition of a musical pattern is a set of ontime pitch pairs that are repeated at least once within a composition [1, 4]. However, most research in MIR struggles to reach a general consensus on what is a musical pattern [18]. Some definitions present patterns as a function of multiple attributes (duration, contour, harmony, etc.) [16]. Some researchers have presented drastically different definitions where patterns do not need to be repeated but must have musical *meaning* to create musical statements and to construct a musical syntax [12].

The research presented by [18] suggests that patterns can be defined as variations on an initial musical element, where variations consist of repetitions with modifications to one or more of its attributes (e.g., pitch, duration). A recognition system for music must incorporate perceptual features for successful operation [25]. The work done by this research, influenced by the above definitions as well as those given by various musicians will consider a more perceptual definition of a musical pattern. We consider every ordered sequence of notes and pauses that occur at least twice in a composition, either identically or with variations, as a pattern. Two sequence occurrences that present artistic variations are considered to be repetitions of the same pattern when untrained human listeners can perceptually agree on their common origin/ equivalence/ similarity.

4.1 Examples of pattern variations

To identify such variations, recordings were obtained from musicians of various musical backgrounds. The dataset DoMP contains a total of 4000 patterns of which 3600 contain artistic variations. Upon careful inspection of DoMP, it was discovered that the recorded repeating patterns possessed the following characteristics:

(1) *The average length of a musical pattern is 20.45 notes and pauses;*
(2) *The average deviation in amplitude is 9.24 MIDI velocity;*
(3) *The average deviation in MIDI tick time time is 77.64;*
(4) *The average number of notes added to a musical pattern is 4.84;*
(5) *The average number of notes removed from a musical pattern is 0.69;*
(6) *The average number of extra pauses added to a pattern is 1.93;*
(7) *The average number of pauses removed from a pattern is 0.33;*
(8) *On average, 9% of the notes in a pattern contain pitch bends;*
(9) *The average number of pitch bends added as extra is 3.65;*
(10) *On average, 956 of the pattern variations had added pitch-bends, when the ground-truth did not;*
(11) *An average of 1.58 of the added notes were chromatic passing notes in between two notes originally one tone apart.*

This understanding was preserved while designing the algorithms to detect patterns in musical streams. For instance, it is evident that approximately 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 statistics above show that extra notes being added is more common than notes being removed, and should be given a higher priority. Similarly, while it is common for extra pauses to be added, it is very rare for existing pauses to be removed. While the inclusion of pitch bends to add modulations such as vibrato is very common, using the pitch bend to reach the next note is less common. As these traits are not localized to a single musician, but the averages of many, this shows that they are adequate to develop an algorithm that is generalized to work with most live performances.

5 Experimental Setup and Evaluation

In this study, we present two approaches for detecting patterns in an incoming music stream. The first method is based on Dynamic Time Warping (DTW) - a traditional Digital Signal Processing method to compute the similarity between two time series. The second method is based on a Recurrent Neural Network, a type of artificial neural network specifically designed for sequential data.

5.1 Dynamic Time Warping

Dynamic Time Warping (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, or the distance 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.

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

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

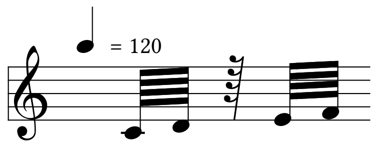
where an alignment path π is a sequence of K index pairs $((i_0, j_0), (i_1, j_1), ..., (i_{K-1}, j_{K-1})$, which represents the shortest possible path between x and x', and $A(x, x')$ is the set of all possible paths.

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: $i_{k-1} \le i_k \le i_{k-1} + 1$, and $j_{k-1} \le j_k \le j_{k-1} + 1$.

For the DTW approach, we first extract a sub-sequence of notes from the incoming music through a windowing function. The size of the window will be dependent on the lengths of the patterns that are to be recognized. As explained in Section 3, a musician may add an average of 24% of additional notes and pauses. Hence, the size of the window function was set to be approximately 20% larger than the length of the longest pattern.

As presented in our previous works [27, 28], the MIDI data is converted to a matrix-based representation for the DTW approach where the pitch and duration of each note is used to represent it.



$$\begin{bmatrix} 60 & 62 & 0 & 64 & 65 \\ 60 & 60 & 60 & 60 & 60 \end{bmatrix}$$

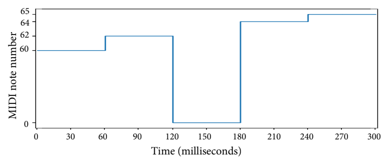


Figure 3: The first four notes of the C major scale *(top)*, and the corresponding matrix-based representation with pitch and duration *(middle)*, and sampled sequence with 30ms sampling frequency *(bottom)*.

Upon receiving each MIDI event, we scan for gradient changes in pitch in order to determine changes between notes and pauses. Whenever a gradient change is detected i.e., the transition between a *note on* message and a *note off* message, or receiving a *pitch bend* message, we consider it as a new event. Fig 3 shows the first four notes of a C-major scale with a pause, and its matrix based representation.

The pitch is an integer in the range 0–127. However, pitch bends are denoted using a separate *pitchwheel* message which lies in the range 0–16383 [2]. Although this range of values could be mapped to a pitch change of any interval in a synthesizer, the common practice is to use a pitch bend of ±2 semitones [2]. We work under this assumption and denote the pitch of a note as a floating point value where a unit pitchwheel increment is considered as a $^1/_{8192}$ increment in pitch. As this system is mainly developed with live improvisational usage in mind, we use the default MIDI tempo of 120 BPM.

A sub-sequence is obtained upon receiving each new event - may it be a note, pause, or a pitch-bend. The DTW between each ground-truth and the sub-sequence is then computed to measure similarity. The MIDI notes are converted to the matrix-based representation format consisting of a pitch vector and a duration vector (ref fig. 3), which requires the use of a multidimensional DTW to obtain a similarity.

This multidimensional DTW may be achieved by a singular warping path across all dimensions, or by merged warping paths of each individual dimension [26, 35]. We have chosen the latter, as it enables us to control the *contribution* of each attribute towards the final similarity measure. This step is crucial in our application, as we have identified that some attributes are more important to the definition of a pattern than others (refer to Sections 3 and 4). As evident by the dataset recorded by real musicians, and as stated in the generally accepted definition of a pattern [1, 4], the pitches of notes are a critical factor and should be given more importance. While the duration of notes do play a part in defining a pattern, the data we collected suggest that it could be given a *lesser* importance.

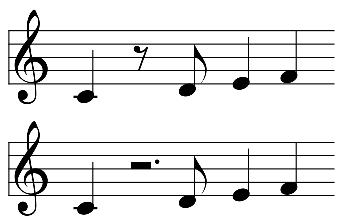


Figure 4: C, D, E, F notes with a short pause (top). The same notes, but with a significantly larger pause (bottom). Due to the longer pause, the two patterns will not seem perceptually equivalent despite the pitches being identical.

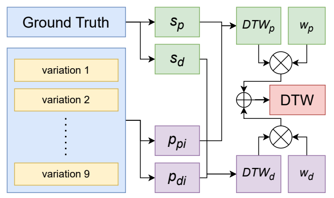


Figure 5: A block diagram of the DTW algorithm.

Refer to Fig. 4 for an example of this scenario, where significant changes in some *less-important* attributes may alter the perceptual definition.

Hence, we have implemented a weighted sum to obtain the final metric. The algorithm splits the sequence to a pitch vector s_p and a duration vector s_d. As shown in Fig. 5, a 1-dimensional DTW for pitch and duration (DTW_p and DTW_d) is obtained between each pitch and duration vector in the sequence (s_p and d_d), and the corresponding pitch and duration vectors of a pattern i in a list of patterns (p_{pi} and p_{di}), to obtain DTW_p and DTW_d. The final metric is obtained via a weighted summation of DTW_p and DTW_d, where w_p and w_d are the weights for pitch and duration respectively. The weights were empirically identified to obtain the highest accuracy.

The DTW method was evaluated to measure its accuracy when computing similarity for increasing numbers of patterns. For each musician, the dataset contains 10 ground truths, as well as 90 variations. 10 rounds of experiments were conducted for each musician where the first evaluation had a single pattern, the second had two patterns, and so on. For each pattern i, the final metric was obtained as: $DTW = DTW_p \times w_p + DTW_d \times w_d$.

5.2 Recurrent Neural Network (RNN)

A Recurrent Neural Network (RNN) is a subclass of artificial neural networks where nodes are interconnected, allowing the output of nodes to have an impact on subsequent nodes. Due to their capabilities of working with synchronous and temporal data, RNNs have many applications such as language translation, time series prediction and analysis, and well as speech synthesis.

The RNN should be able to detect if a sequence of musical notes and pauses is *similar* to one of the pre-defined patterns and produce a statistical estimate of the similarity. To achieve this, we use an RNN classification, where the number of output classes is equal to the number of patterns, along with an additional class to denote not-a-pattern.

Several different RNN configurations exist, each with their own merits and drawbacks. For our approach, we empirically identified that a stacked configuration, consisting of a fully connected RNN layer, a GRU layer, and a LSTM layer yielded the best accuracy.

A Fully Connected RNN consists of an architecture where each node is connected to every node in the next layer. This comprehensive connectivity allows information to flow between all nodes. Fully connected RNNs can capture intricate relationships present in sequential data more effectively than simpler architectures such as Vanilla RNNs. However, due to their complex structure, fully connected RNNs may require more training data and computational resources.

The long-short term memory (LSTM) networks involve a more complex memory which allows it to retain information over long sequences. LSTMs consist of a hidden state and a cell state, which contain the input at the current time step along with the information learned from previous time steps. The LSTM has three gates as illustrated by fig 6. The input gate regulates the flow of new information into the cell state, the forget gate controls the information to be discarded, and the output gate controls the flow of information from the cell state to the hidden state. This architecture allows the LSTM layer to capture long-term dependencies in sequential data

The gated recurrent unit (GRU) layer is also able to overcome the vanishing gradient through its update gate and reset gate, which determines the degree of past information forwarded to the future and the amount of past information to discard respectively. The GRU also includes a candidate hidden state as well as a final hidden state, which allows it to selectively update its hidden state at each time step, making it well-suited for tasks with sequential information.

The RNN method uses a discrete time series representation of the input data. This is achieved by sampling the MIDI input every 30 milliseconds to obtain a sub-sequence with equal temporal spacing between elements. We used intervals of 30 ms as this is approximately the temporal resolution of the human hearing system to distinguish two sequential sound events [21]. Similar to the DTW method, the pitch is denoted as a floating point value where a unit increase in MIDI pitchwheel corresponds to $1/8192$ increment in pitch. Fig. 3 shows an example for this time-series representation, where the first four notes of the C major scale in standard musical format and the sampled time series.

Creation of the synthetic training dataset

A primary requirement for any accurate deep learning model is the availability of sufficient training data that correspond to possible variations and fluctuations of the input. In an approach such as ours, the training set must adequately represent all variations that musicians may perform as described in section 4. It is impractical for a musician to record numerous versions of each single pattern. Hence we analyzed the pattern variations present in the dataset to identify the most commonly applied variations, and apply them to create a synthetic training dataset of expressive and melodic variations given a single ground-truth.

These variations are inspired by those present in the recorded dataset. Upon extensive analysis of the dataset, we identified several expressive variations that were common across most musicians. Some examples for said variations are:

- 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;
- 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.

Such common expressive variations are used as rules when creating the synthetic dataset. Some examples of the applied rules are:

- Changing the duration of arbitrary notes and pauses by a constant factor, so as the total duration of the melody remains unchanged.
- Doubling notes: It was identified that approximately 23.6% notes may be added as extra. An arbitrary number of notes between 0 and 23.6% are doubled, while halving their duration.
- Removal of notes: It was identified that approximately 3.3% notes may be removed. An arbitrary number of notes between 0 and 3.3% are removed. The durations are left unchanged, and the removed duration is added to other arbitrary notes/pauses.
- Pitch bends: For an arbitrary number of notes between 0 and 9%, where each note and the subsequent note are less than 2 semitones apart, a pitch bend is applied to the first note to gracefully reach the pitch of the subsequent note, which is removed and its duration added to the first note.
- Chromatic notes: If a note and the subsequent note are exactly 2 semitones apart, adding the middle note.

There have been studies that apply rule-based modulation to mimic certain styles of music, where the rules may be strict [3]. However, our approach tries to generalize expressive variations that has the potential to work with most musicians' playing styles. Longer musical patterns have the potential for more expressive variation, and vice versa. It was empirically identified that approximately 10000 such variations per ground-truth were sufficient.

During training a neural network, it is crucial to have all training data in an equal size. As the synthesized dataset contains different sized variations, we pad the training dataset to fit the length of the longest variation present.

As the task is a relatively simple time series classification, and as the model needs to be light enough to run on an embedded system in real time, the RNN layer consists of 64 neurons, the GRU layer consists of 32 neurons, and the LSTM layer consists of 64 neurons. Furthermore, we use a masking layer to disregard any padded inputs in the training dataset. The training dataset is also sampled at 30 ms increments to maintain consistency between training data and actual input.

The evaluation of the RNN was kept as close as possible to the evaluation of the DTW to maintain a fair comparison. 10 groups of musical patterns, each with 1 to 10 patterns were used. For each case, the RNN was trained with synthetically created variations as discussed, and evaluated against the *actual* pattern variations

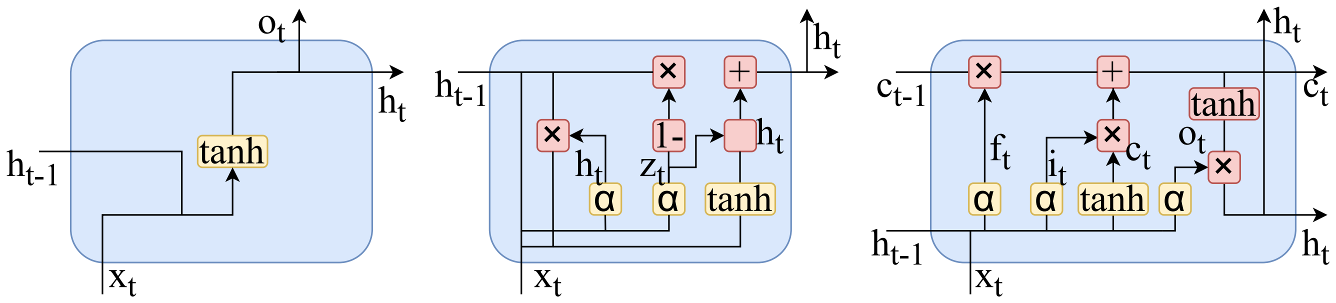


Figure 6: A traditional RNN layer *(left)*; a GRU layer *(middle)* with a reset gate r_t and an update gate z_t; and a LSTM layer *(right)* with a forget gate f_t, input gate i_t, and output gate o_t. At each timestep t: x_t, o_t, h_t, and c_t are the input, output, hidden state, and intermediate cell state, α and $tanh$ are the sigmoid and tanh activation functions respectively.

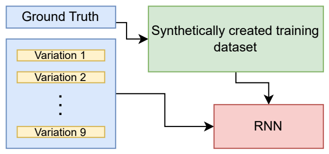


Figure 7: A Block diagram of evaluation of the RNN algorithm.

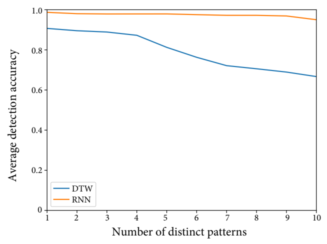


Figure 8: Average detection accuracy against number of patterns.

recorded by the musician. Fig. 7 shows a block diagram of the evaluation.

6 Results

The evaluations were conducted in a way that would warrant a fair comparison between DTW and RNN. Patterns were grouped together to assess the detection accuracy when multiple patterns are present. RNN and DTW systems were evaluated on their ability in detecting a single pattern up to detecting 10 distinct patterns.

The MIDI pitch and duration of each musical note was used to represent it for computations. The DTW method uses the MIDI data represented through a matrix-based format, and the RNN uses a discrete time-series obtained by sampling the MIDI data. Fig. 8 shows the average accuracies of the DTW as compared to the RNN method for groups of patterns ranging from 1-10.

Upon conducting evaluations where the music was represented by pitch, velocity, and duration [27, 28], a significant reduction of

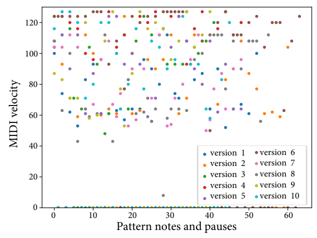


Figure 9: MIDI velocity values for 9 versions of a pattern in the dataset. The plot indicates the velocity value of each note in the pattern, and 0 velocity represents the pauses. The plot shows that velocity is an unreliable factor given its inconsistency and high variance.

detection accuracy was observed as illustrated in Fig 9. Upon investigation of the dataset, we noticed that changes in velocity were very common among pattern repetitions (refer fig. 11). As there is a high variance in velocity, even among repetitions of the same patterns, including it in computation will merely inhibit the systems accuracy. Hence we discarded velocity from our evaluations.

Fig. 10 shows the DTW results for 4 ground truths and their variations individually. Each subplot represents a single ground truth along with the dtw distances against the ground truth for its recorded variations. The first index of the x-axis is the ground truth, thereby resulting in a DTW distance of zero with itself. The grouping of musical patterns was done to simulate a hypothetical real-world case, where a musician would select several iconic musical phrases that they would include in their performance in order to control peripheral devices in real-time.

As our algorithms are designed to work in real time, a low latency is a primary requirement. Latency values for the RNN method when classifying 1 - 10 patterns a is shown by table 1. These latency values show the system's suitability to work in real time.

All evaluations were done with Python using the Keras API for TensorFlow. The training was done on a computer with an *Intel(R) Core(TM) i9-10940X CPU @ 3.30GHz* using *NVIDIA GeForce RTX 4090* and *NVIDIA GeForce RTX 3090* GPUs. The evaluation of the

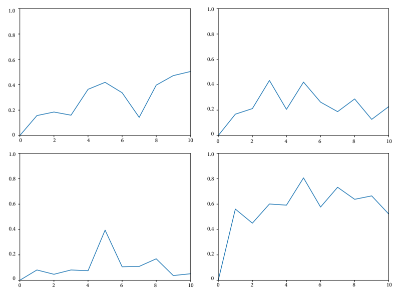


Figure 10: dtw distances for pattern variations. X-axis represents each variation, while the y-axis represents the DTW value for said variation.

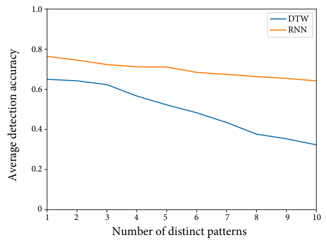


Figure 11: Average detection accuracy against number of patterns when using pitch, velocity, and amplitude.

algorithms was done on a Laptop with an *Intel(R) Core(TM) i7-11800H @ 2.30GHz*. However, we have implemented them within an embedded system for real-time inference.

7 Discussion and Conclusion

In this paper, we presented a comparison between Dynamic Time Warping and Recurrent Neural network based methods for musical pattern detection in real-time. The experimental results show that the RNN performs better across the board, especially when increasing the number of distinct patterns to recognize. This accuracy may be attributed to the RNN's ability to process sequential data, and its ability to capture long-term dependencies in time series data, learn complex, non-linear relationships between input and output variables, as well as their ability to handle inputs of different lengths - which is a major characteristic required for pattern detection tasks such as ours.

The adequacy of the training dataset may also be attributed to the high accuracy of the RNN. The synthetically created pattern variations, inspired by those present in the dataset, act as good representations of real inputs as evident by the results. With this synthetically created training dataset, we overcome a main drawback in any neural network-based method - the requirement of sufficient data for training.

However, a drawback in the RNN method is the high computational power required for training. For our evaluations, the training was done on a high performing remote server. The inference, on the other hand requires minimal power. Our evaluations, which were done on a standard laptop computer revealed that the memory and CPU consumption during inference in minimal, and it can be done on an embedded computing device.

Our contributions by this study are: an RNN based algorithm to detect musical patterns in real-time; a novel dataset of musical patterns and expressive and melodic variations, and a model to create a synthetic training dataset to train the RNN model. While the evaluation of the RNN-based method produced good results, there is ample room for improvement. In future work we plan to extend this study to work with polyphony - where a musician may utilize patterns with multiple notes at a given time instant. The extension to work with polyphony will also allow for multi-instrument patterns to be defined. All these features could be exploited by composers and performers to explore novel artistic forms, which are enabled by the smart musical instruments and the IoMusT paradigm.

Furthermore, the algorithm presented by this paper resulted in the development of a prototype smart electric guitar[4], with monophonic pattern detection capabilities. The prototype smart electric guitar demonstrates RNN's ability to detect patterns in real-time, using minimal computational resources. The RNN algorithm has been implemented within an embedded computing system, mounted on the electric guitar body. Furthermore, it has the capability to control peripheral devices wirelessly. We plan to further develop this musical instrument to work with polyphony, as well as audio signals instead of MIDI. Our hope is that developments such as this in the field of IoMusT will spur the creativity of musicians in incorporating multisensory components (e.g., visual, tactile, olfactory) into their musical performances.

References

[1] 2018. *2017:Discovery of Repeated Themes & Sections.* Retrieved Aug 12, 2024 from https://www.music-ir.org/mirex/wiki/2017:Discovery_of_Repeated_Themes_%26_Sections
[2] 2024. *Official MIDI Specifications: General MIDI 1.* Retrieved Aug 12, 2024 from https://midi.org/general-midi
[3] M. Amerotti, S. Benford, B. Sturm, , and C. Vear. 2023. A Live Performance Rule System Informed by Irish Traditional Dance Music. In *Proceedings of the International Symposium on Computer Music Multidisciplinary Research.*
[4] A. Arronte Alvarez and F. Gómez. 2021. Motivic Pattern Classification of Music Audio Signals Combining Residual and LSTM Networks. *International Journal of Interactive Multimedia and Artificial Intelligence* (05 2021).
[5] S. Chakrabarty, R. Islam, E. Pricop, and H. Sarma. 2022. An Approach to Discover Similar Musical Patterns. *IEEE Access* 10 (2022), 47322–47339.
[6] T. P. Chen and L. Su. 2017. Discovery of repeated themes and sections with patern clustering. In *Presented at the 18th International Society for Music Information Retrieval Conference.*
[7] T. Collins, J. Thurlow, R. Laney, A. Willis, and P. Garthwaite. 2010. A comparative evaluation of algorithms for discovering translational patterns in Baroque keyboard works. In *Proceedings of the International Symposium on Music Information Retrieval.*
[8] D. Conklin and C. Anagnostopoulou. 2001. Representation and Discovery of Multiple Viewpoint Patterns. In *Proceedings of the 2001 International Computer Music Conference.* 479–485.
[9] R. Dannenberg and N. Hu. 2002. Linear time for discovering non-trivial repeating patterns in music databases. In *ISMIR 2002 Conference Proceedings: Third International Conference on Music Information Retrieval.* 63–70.
[10] T. de Reuse and I. Fujinaga. 2019. Pattern Clustering in Monophonic Music by Learning a Non-Linear Embedding From Human Annotations. In *Proceedings of the 20th International Society for Music Information Retrieval Conference.* 761–768.
[11] C. Finkensiep, M. Neuwirth, and M. Rohrmeier. 2018. Generalized Skipgrams for Pattern Discovery in Polyphonic Streams. In *Proceedings of the 19th International Society for Music Information Retrieval Conference.* 23–27.
[12] F Gómez, M Tizón, A Arronte Alvarez, and V Padilla. 2022. Rhetorical Pattern Finding. *International Journal of Interactive Multimedia and Artificial Intelligence* (11 2022).

[4]https://hotlicksvst.github.io/

Table 1: RNN average latency values for inference.

Number of patterns	1	2	3	4	5	6	7	8	9	10
Latency (milliseconds)	4.78	4.87	4.75	4.67	4.97	5.15	4.83	4.69	4.74	4.77

[13] J. L. Hsu, C. Liu, and A. Chen. 2001. Discovering nontrivial repeating patterns in music data. *IEEE Trans. Multim.* 3 (2001), 311–325.

[14] I. Knopke and F. Jürgensen. 2009. A System for Identifying Common Melodic Phrases in the Masses of Palestrina. *Journal of New Music Research* 38, 2 (2009), 171–181.

[15] Q. Kong, Y. Cao, T. Iqbal, Y. Wang, W. Wang, and M. D. Plumbley. 2020. PANNs: Large-Scale Pretrained Audio Neural Networks for Audio Pattern Recognition. *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 28 (2020), 2880–2894.

[16] O. Lartillot and P. Toiviainen. 2007. Motivic matching strategies for automated pattern extraction. *Musicae Scientiae* 11, 1_suppl (2007), 281–314.

[17] C. Meek. 2003. Automatic Thematic Extractor. *Journal of Intelligent Information Systems* 21 (2003), 9–33.

[18] O. Melkonian, I. Ren, W. Swierstra, and A. Volk. 2019. What Constitutes a Musical Pattern?. In *Proceedings of the 7th ACM SIGPLAN International Workshop on Functional Art, Music, Modeling, and Design* (Berlin, Germany). Association for Computing Machinery, New York, NY, USA, 95–105.

[19] D. Meredith. 2017. Using SIATECCOMPRESS to discover repeated themes and sections in polyphonic music. In *Presented at the 18th International Society for Music Information Retrieval Conference.*

[20] D. Meredith, K. Lemström, and G. A. Wiggins. 2002. Algorithms for discovering repeated patterns in multidimensional representations of polyphonic music. *Journal of New Music Research* 31, 4 (2002), 321–345.

[21] B. C. J. Moore. 2012. *An introduction to the psychology of hearing.* Brill.

[22] O. Nieto and M. M. Farbood. 2017. Music segmentation techniques and greedy path finder algorithm to discover musical patterns. In *Presented at the 18th International Society for Music Information Retrieval Conference.*

[23] M. Pesek, A. Leonardis, and M. Marolt. 2017. SYMCHMMERGE: an extension to the compositional hierarchial model for pattern discovery in symbolic music representations. In *Presented at the 18th International Society for Music Information Retrieval Conference.*

[24] I. Y. Ren, H. V. Koops, A. Volk, and W. Swierstra. 2017. In Search of the Consensus Among Musical Pattern Discovery Algorithms. In *Proceedings of the 18th International Society for Music Information Retrieval Conference.* 23,27.

[25] I. Shmulevich, O. Yli-Harja, E. Coyle, D. Povel, and K. Lemström. 2001. Perceptual Issues in Music Pattern Recognition: Complexity of Rhythm and Key Finding. *Computers and the Humanities* 35, 1 (2001), 23–35.

[26] M. Shokoohi-Yekta, B. Hu, H. Jin, J. Wang, and E. Keogh. 2017. Generalizing DTW to the multi-dimensional case requires an adaptive approach. *Data Mining and Knowledge Discovery* 31 (01 2017). https://doi.org/10.1007/s10618-016-0455-0

[27] N. Silva, C. Fischione, and L. Turchet. 2020. Towards Real-Time Detection of Symbolic Musical Patterns: Probabilistic vs. Deterministic Methods. In *2020 27th Conference of Open Innovations Association (FRUCT).* 238–246.

[28] N. Silva and L. Turchet. 2022. A Structural Similarity Index bsed method to detect symbolic monophonic patterns in real-time. In *Proceedings of the 25th International Conference on Digital Audio Effects.* 161–168.

[29] M. Thomas, Y. V. S. Murthy, and S. G. Koolagudi. 2016. Detection of largest possible repeated patterns in Indian audio songs using spectral features. In *2016 IEEE Canadian Conference on Electrical and Computer Engineering.* 1–5.

[30] L. Turchet. 2019. Smart Musical Instruments: vision, design principles, and future directions. *IEEE Access* 7 (2019), 8944–8963.

[31] L. Turchet, C. Fischione, G. Essl, D. Keller, and M. Barthet. 2018. Internet of Musical Things: Vision and Challenges. *IEEE Access* 6 (2018), 61994–62017.

[32] P. van Kranenburg, A. Volk, and F. Wiering. 2013. A Comparison between Global and Local Features for Computational Classification of Folk Song Melodies. *Journal of New Music Research* 42, 1 (2013), 1–18.

[33] A. Volk and P. van Kranenburg. 2012. Melodic similarity among folk songs: An annotation study on similarity-based categorization in music. *Musicae Scientiae* 16, 3 (2012), 317–339. https://doi.org/10.1177/1029864912448329

[34] C. I. Wang, J. Hsu, and S. Dubnov. 2015. Music Pattern Discovery with Variable Markov Oracle: A Unified Approach to Symbolic and Audio Representations. In *Proceedings of the 16th International Society for Music Information Retrieval Conference.*

[35] M. Wöllmer, M. Al-Hames, F. Eyben, B. Schuller, and G. Rigoll. 2009. A multidimensional dynamic time warping algorithm for efficient multimodal fusion of asynchronous data streams. *Neurocomputing* 73, 1 (2009), 366–380.