

# From Neural Rhythm Classification to Embodied Sense-Making: Strummin’ Gym as a Browser-Based IoMusT Environment for Guitar Strumming Practice

Marco Pennese

Conservatory of Music “G. Puccini”  
La Spezia, Italy  
marco.pennese@steffani.it

Stefano Giacomelli

DISIM dpt., University of L’Aquila  
L’Aquila, Italy  
stefano.giacomelli@graduate.univaq.it

Claudia Rinaldi

CNIT University of L’Aquila Research Unit  
L’Aquila, Italy  
claudia.rinaldi@univaq.it

**Abstract**—Pop-rock guitar accompaniment relies on rhythmic, gestural, and harmonic knowledge that common lyrics-and-chords resources represent only partially. Such materials typically encode chord symbols while leaving pulse, subdivision, accentuation, strumming motion, and chord duration implicit. Framed within music education applications for the Internet of Musical Things, this paper presents Strummin’ Gym, a web-based environment for beginner guitar learning that transforms audio-informed rhythmic analysis into editable multimodal practice configurations. The Strumming Setup integrates two neural models: StrumKANet, which suggests one of three pedagogically defined strumming patterns, and BeatNet, which estimates BPM and a reference beat grid. The backend preliminarily validates the alignment among the suggested pattern, tempo, and beat structure, enabling the user to audition the inferred strumming pattern synchronized with the target audio track. Each parameter remains explicitly editable, allowing the learner or teacher to correct, refine, or customize the AI-generated configuration.

A dedicated Chord Progression Builder extends the same parameters to the construction of executable rhythmic-harmonic materials. Users can assign each chord an exact duration expressed as repetitions of the selected or customized strumming pattern, thereby converting a conventional chord sequence into a performable practice object. Rather than assessing performance, Strummin’ Gym is designed to support rhythmic-gestural understanding in both autonomous and teacher-led study.

**Index Terms**—Internet of Musical Things, music education, guitar learning, strumming practice, embodied interaction, music information retrieval, neural rhythm analysis, beat tracking, downbeat tracking, synchronization, StrumKAN, BeatNet

## I. INTRODUCTION

**P**OP-ROCK guitar accompaniment is one of the most widespread forms of instrumental practice, yet the materials most commonly available to beginners represent it only partially. Strummin’ Gym specifically targets the common lyrics-and-chords format, in which chord symbols are aligned with song text while rhythmic, metrical, and durational information is typically left implicit. More detailed symbolic representations, such as lead sheets or standard notation, fall outside the scope of the system. In strumming-based practice, the relevant learning problem is not only *which* chords to play, but *how* they are articulated: pulse, meter, subdivision, accentuation, sounding and silent strokes, continuous down-up motion, and chord duration.

This gap is visible in the broader ecosystem of guitar-learning technology. Digital resources have long supported access to chords and instructional content [1], [2], and large-scale platforms such as Ultimate Guitar allow strumming-pattern annotations to be attached to chord charts [3], [4]. This confirms that harmonic information alone is recognized as insufficient for pop-rock accompaniment. However, a strumming pattern attached to a chart does not make it fully executable: the chart specifies *which* pattern to use, but not *how long* each chord should last relative to that pattern – information a learner unfamiliar with the song does not have.

This paper presents Strummin’ Gym, a browser-based environment for guitar strumming practice that closes this gap. The system builds on our previous StrumKAN framework, which modeled recurrent pop-rock strumming templates as a mid-level MIR classification task [5]. Rather than pursuing further classification accuracy, the present contribution investigates how neural rhythm classification can be operationalized as a rhythm-gesture scaffold: an Internet-accessible practice environment integrating rhythmic analysis, beat/downbeat tracking, synchronization, metronomic control, visual feedback, and chord-duration logic into a single workflow.

From the user’s point of view, the GUI, shown in Fig. 1, is organized around a clear distinction between Practice Mode and Song Mode. Practice Mode allows learners and teachers to construct practice material without relying on a specific audio reference. Song Mode introduces a specific track and possible AI-based suggestions concerning execution parameters. This visible dichotomy is preserved in the system design, but the paper argues that the pedagogical value of the interface depends on how these two operational contexts make rhythmic, gestural, and harmonic information mutually decodable.

The central outcome of this workflow is a fully executable chord chart.

The theoretical framing draws on recent IoMusT pedagogy. Nijs and Turchet propose a design framework for IoMusT applications organized around embodied sense-making, non-linearity, participatory sense-making, privacy/security, and accessibility [6], a lens compatible with strumming practice, where musical understanding is grounded in bodily alignment

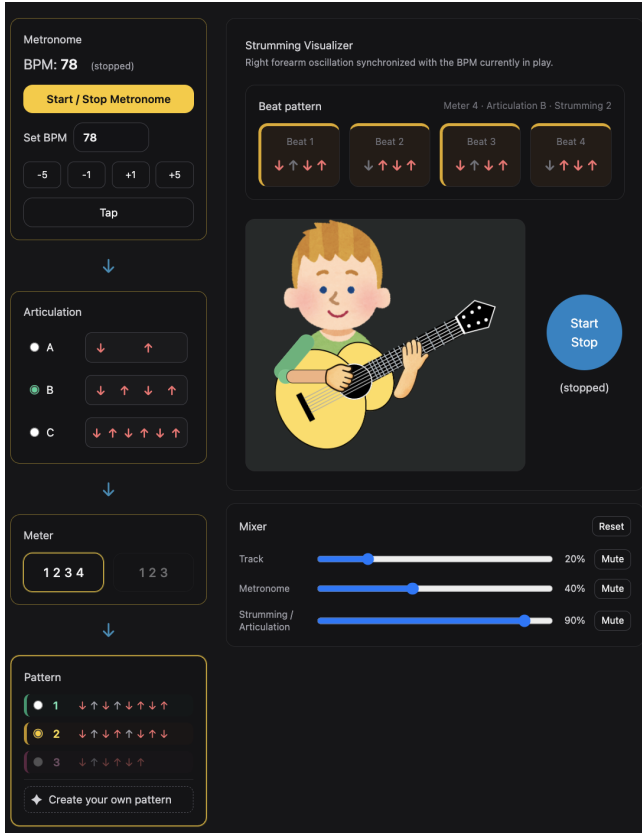


Fig. 1. Screenshot of Strummin' Gym browser interface showing execution parameters and 2D avatar (with available mixer management).

with pulse, meter, and gesture. This is consistent with embodied and enactive accounts of cognition, in which understanding arises from action and environmental coupling rather than abstract information processing alone [7].

Strummin' Gym is deliberately not an automatic assessment platform. Recent surveys on AI in music education discuss personalization, feedback, data use, teacher agency, and automated grading as open and contested issues [8]–[10]. Rather than adjudicate these debates, the system focuses on practice scaffolding: transforming MIR and neural outputs into editable, multimodal cues that support rhythmic-gestural understanding before and during performance.

This paper makes three contributions. First, it situates Strummin' Gym within IoMusT framework and embodied sense-making. Second, it presents a network architecture that separates server-side MIR/neural analysis from low-latency browser-based practice. Third, it describes how StrumKAN-based pattern recognition, BeatNet-based beat/downbeat tracking, and a repeat-based chord-duration logic combine into executable practice configurations.

The remainder of the paper is organized as follows. Section II situates the work within IoMusT and embodied sense-making. Section III discusses the transition from StrumKAN to Strummin' Gym. Section IV presents the system architecture and prototype implementation. Section V describes

the rhythm-gesture workflow and the didactic organization of the interface. Section VI discusses design implications, limitations, and future work, and Section VII concludes.

## II. RELATED WORK: IOMUST AND EMBODIED SENSE-MAKING

The Internet of Musical Things (IoMusT) provides a framework for systems in which sensing, processing, exchange, and actuation of music-related data support new forms of musical interaction. In education, this perspective is relevant when networked technology moves beyond content delivery to actively mediate musical action and learning. Nijs and Turchet's framework for IoMusT and children offers a design lens for pedagogically grounded musical technologies, although it targets early-childhood education specifically, whereas Strummin' Gym does not [6]. Digital tools for guitar and music-theory instruction more broadly have followed a similar trajectory, from static reference material toward interactive, feedback-driven platforms [1], [2], [11].

Embodied sense-making is central to the present design. Embodied and enactive accounts of cognition hold that understanding emerges from the coupling of perception, movement, and environment [7] – a view well suited to instrumental learning, where musical knowledge is simultaneously symbolic and gestural, temporal, and situated. Instrumental pedagogy makes a parallel argument for the operational, bodily dimension of learning, in which technical actions, musical intention, and didactic mediation are progressively coordinated [12], while Bruner's account of education as structured, revisitable understanding supports organizing complex musical knowledge into accessible, iterable forms [13].

Beat induction links music cognition directly to instrumental action. Honing describes it as the fundamental capacity to perceive a regular pulse and synchronize with it [14]; for strumming practice this capacity must be *enacted* through repeated down-up motion, selective activation of sounding/silent strokes, and bodily alignment with the perceived meter. Strummin' Gym addresses this by translating rhythmic cues into synchronized visual, auditory, and gestural feedback via Hugo (Helping Users get the Groove On), a 2D strumming avatar that provides an audio-visual emulation of the gesture and its sonic result. The system requires no dedicated sensing hardware, exposing music-learning services entirely through the browser.

The online dimension also introduces ethical constraints. IoMusT research argues that networked musical technologies require explicit ethical reflection given their data practices, human participants, and social impact [15], a concern reinforced by policy guidelines on AI and data in education [16]. For Strummin' Gym this implies designing audio upload, AI suggestions, and practice configurations around transparency, data minimization, user autonomy, and copyright-safe content.

## III. FROM STRUMKAN TO STRUMMIN' GYM

Our previous work introduced StrumKAN [5], a neural framework for automatic strumming-pattern recognition in

pop-rock music, formulating pattern classification as a mid-level MIR task over a limited set of recurring rhythmic-gestural templates – following pedagogical models such as Perlak’s [17]. The present work shifts focus from model validation to pedagogical deployment: a predicted strumming category is not treated as a final answer, but becomes useful only once linked to tempo, meter, articulation, synchronization, and chord duration. The present work does not re-evaluate the StrumKANet architecture or claim an intrinsic advantage of its KAN head over alternative classifiers; it reuses the previously selected model as a rhythm-classification component within the new practice environment. The classifier output is thus reframed as a *practice cue*, an interpretable, editable suggestion that supports listening, experimentation, and teacher-guided correction rather than an authoritative label.

This editable status is a pedagogical choice, not only a technical safeguard. Because neural predictions are inevitably fallible, the learner is invited to test the AI-generated setup against listening, bodily execution, and teacher guidance, asking whether the proposed pattern, tempo, or synchronization point matches the perceived musical situation. The suggestion thus functions as a reflective cue rather than ground truth.

The distinction matters because chord/tab ecosystems already offer lightweight strumming annotations: Ultimate Guitar charts may include one or more community-supplied strumming patterns [3], [4]. Such annotations are useful as symbolic references, but remain weakly connected to the temporal and gestural conditions of practice – a chart rarely specifies whether a chord should last one, two, or four pattern cycles, so a learner unfamiliar with the song still lacks the durational information needed to perform it coherently.

Strummin’ Gym addresses this by replacing static annotation with an *executable practice configuration*: the user selects not only chords and a strumming pattern, but a complete execution setup including BPM, articulation, meter, pattern, and the number of pattern repetitions assigned to each chord. The chord progression thus becomes a rhythmic-harmonic object in which each chord carries an explicit duration, playable coherently with metronome, strumming audio, and visual feedback.

Realizing this shift requires combining pattern recognition with temporal synchronization. Strummin’ Gym therefore pairs StrumKAN-based pattern detection with BeatNet, an on-line system for joint beat, downbeat, and meter tracking based on a CRNN architecture with particle filtering [18]. In this context BeatNet does not produce a standalone transcription; it provides the temporal anchors needed to align a predicted or user-selected pattern with a reference audio track inside the browser.

The transition from StrumKAN to Strummin’ Gym is therefore both technical and pedagogical: technically, it requires moving from offline classification to an architecture that separates server-side analysis from browser-side interaction; pedagogically, it requires moving from classification accuracy to embodied sense-making, i.e., whether the system helps

the learner understand how a pattern is felt, synchronized, anticipated, and enacted.

#### IV. SYSTEM ARCHITECTURE AND PROTOTYPE IMPLEMENTATION

The proposed architecture separates computationally intensive musical analysis from low-latency interaction, as shown in Fig. 2.

In the current prototype, both the browser frontend and the analysis backend are executed locally on the same host; the client–server separation therefore reflects a logical software architecture rather than a remote network deployment. **Client layer.** The browser hosts the interactive learning environment: Practice Mode, Song Mode, WebAudio-based metronomic scheduling, waveform navigation, beat-pattern rendering, chord-progression playback, and Hugo, the 2D strumming avatar.

**Service/orchestration layer.** A web API manages audio upload, analysis-job creation, result retrieval, and session state. Long-running MIR processes are not executed inside synchronous HTTP requests but delegated to asynchronous workers, because the system must support two distinct temporalities: analysis can be asynchronous, whereas practice interaction must remain responsive.

**Analysis layer.** Audio preprocessing, StrumKAN-based pattern recognition, and BeatNet-based beat/downbeat/meter tracking feed an integration stage that fuses their outputs into a single practice-ready configuration.

**Data layer.** Temporary object storage holds uploaded or rights-cleared audio and analysis outputs; a model registry tracks checkpoints, preprocessing parameters, and versioning. In the current prototype, complete workflow storage, workflow recall, teacher-student sharing, and printable export are not implemented as persistent online features; they are treated as future developments in Section VI.

The central output of the architecture is the *practice configuration object*: pattern, articulation, meter, BPM, synchronization point, beat/downbeat grid, strumming grid, chord sequence, repeats per chord, model versions, and editable AI cues. In the current prototype, this object is returned to the browser and used to drive the interactive practice session. It is therefore a temporary and editable configuration for enacted practice, not yet a persistent, shareable, or recallable learning workflow. A supplementary demonstration video illustrates the current Strummin’ Gym prototype workflow, including AI-assisted rhythmic configuration, synchronized audio-visual practice, and chord-progression authoring.<sup>1</sup>

##### A. Neural Network Sub-Modules

The current prototype integrates two complementary machine-listening Neural Networks: StrumKANet [5], which estimates a pedagogically meaningful strumming pattern from the input audio, and BeatNet [18], which provides the temporal scaffold used to synchronize the track, metronome, and beat-pattern highlighting. The two modules do not produce a

<sup>1</sup><https://www.loom.com/share/d3809bf7a3124d33913cd93487405bd2>

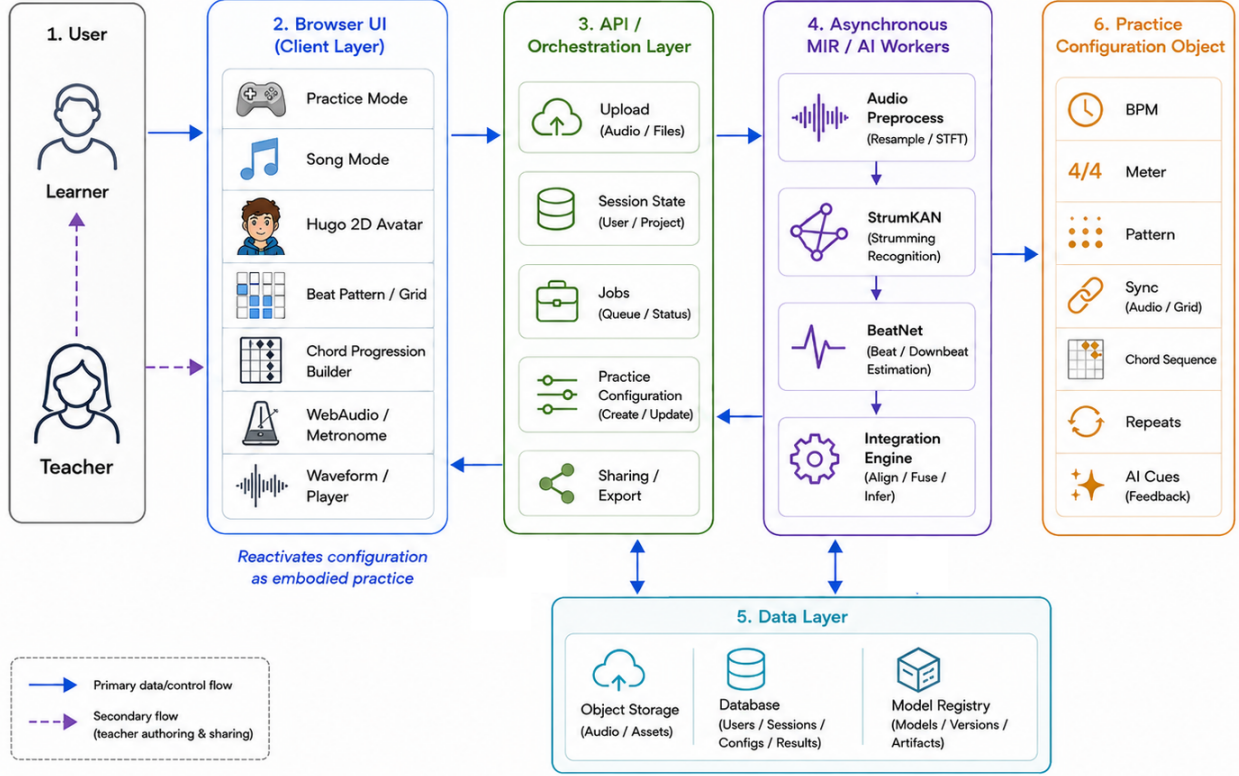


Fig. 2. Network architecture for browser-based strumming practice.

closed automatic transcription; rather, they generate an editable configuration that the learner or teacher can validate, listen to, and modify.

Given an input waveform  $x[n]$ , **StrumKANet** converts the signal to mono, re-sampling at  $f_s = 44.1$  kHz and segments it into non-overlapping 10-s excerpts. No additional segment-level windowing is applied before feature extraction; each complete 10-s excerpt is passed directly to the Rhythm Pattern extractor. For each segment, the implemented pipeline extracts only the *rhythmic patterns* descriptor provided by the TU\_Wien backend [19]: a flattened vector  $\mathbf{r}_i \in \mathbb{R}^{1440}$  representing modulation amplitudes across 24 Bark-scale frequency bands and modulation-frequency bins. The vector is then reshaped and expanded to the tensor format required by the network:  $\mathbf{X}_i \in \mathbb{R}^{1 \times 1 \times 24 \times 60}$ .

The checkpointed StrumKANet model uses the top-performing configuration identified in [5]: a 2D convolutional encoder, followed by a Wavelet Kolmogorov–Arnold classifier. The encoded representation is obtained through 2D convolution, followed by adaptive average pooling and flattening. The PatternKAN head maps the resulting embedding to three pattern logits using Mexican-hat wavelet activations [20]. A representative WavKAN unit is:

$$g(z) = \sum_{k=1}^K \alpha_k \psi\left(\frac{z - \mu_k}{s_k}\right), \quad \psi(u) = (1 - u^2)e^{-u^2/2} \quad (1)$$

where  $\alpha_k$ ,  $\mu_k$ , and  $s_k$  are learnable amplitude, shift, and scale parameters. The segment-level posterior distribution over the three strumming classes is obtained as argmax of a softmax inference.

The track-level pattern suggestion is finally computed by majority voting over all segments:

$$\hat{c} = \min \left\{ c : c \in \arg \max_{q \in \{1,2,3\}} \sum_i \mathbb{I}[\hat{c}_i = q] \right\} \quad (2)$$

where min operator reflects a tie-breaking rule, which selects the lowest label in case of equal votes. The resulting  $\hat{c} \in \{1, 2, 3\}$  is displayed in the GUI as the suggested strumming pattern and mapped to the corresponding beat-pattern panel.

**BeatNet** is used as the temporal-analysis module of the prototype, providing the beat grid, candidate downbeats, local tempo, and meter cues required to synchronize playback, metronome pulses and gesture emulation.

Given the same input waveform  $x[n]$ , the prototype calls the official BeatNet implementation with the selected pre-trained CRNN model, offline processing, and DBN inference [18]. The audio is processed by the internal front-end, which converts the waveform into a frame-wise spectral sequence. Each frame is represented by a log-amplitude time–frequency description projected onto a logarithmically spaced filterbank, and the first-order temporal difference of this representation is concatenated to encode short-term spectral change. This produces the feature sequence provided to the CRNN.

The CRNN first applies convolutional processing along the spectral axis to extract local frequency-pattern information from each frame. The resulting frame embeddings are then passed through recurrent layers, which model the temporal evolution of rhythmic evidence across the audio sequence. A final classification layer produces frame-level activations for beat, downbeat, and non-beat states. In the prototype configuration, these activations are not used directly by the GUI; they are decoded by BeatNet’s offline DBN inference stage, which imposes temporal and metrical consistency on the frame-wise predictions [18].

### B. Backend Post-Processing

The decoded BeatNet output is represented in the backend as an ordered sequence of rhythmic events:

$$\mathcal{V} = \{(t_j, q_j)\}_{j=1}^J, \quad (3)$$

where  $t_j$  is the timestamp of the  $j$ -th decoded event in seconds,  $q_j$  is its estimated metrical position, and  $J$  is the number of decoded events returned by the BeatNet process.

The backend removes invalid timestamps, sorts the events by time, and suppresses duplicate detections closer than 10 ms. After this cleaning stage, the beat grid and candidate downbeat set are defined as:

$$\mathcal{B} = \{t_j\}, \quad \mathcal{D} = \{t_j \in \mathcal{B} \mid q_j = 1\}, \quad (4)$$

where  $\mathcal{B}$  is the BeatNet-derived beat sequence and  $\mathcal{D}$  contains the events decoded as first beats of the bar. The meter estimate  $\hat{m}$  is obtained from the maximum metrical position returned by BeatNet. Values equal to 3 and 4 are retained, while a detected value of 2 is mapped to 4 for compatibility with the GUI meter representation.

The synchronization anchor is selected from the decoded grid by searching for the first downbeat after 15 s. If no downbeat is available, the first beat after the same threshold is used:

$$t_{\text{sync}} = \begin{cases} \min \{t \in \mathcal{D} \mid t \geq 15\}, & \text{if } \{t \in \mathcal{D} \mid t \geq 15\} \neq \emptyset, \\ \min \{t \in \mathcal{B} \mid t \geq 15\}, & \text{if } \{t \in \mathcal{B} \mid t \geq 15\} \neq \emptyset, \\ \min \mathcal{B}, & \text{otherwise.} \end{cases} \quad (5)$$

The last case corresponds to the implemented fallback in which, if no beat is found in the post-15 s candidate window, the backend resets the candidate region and uses the available beat sequence. The selected  $t_{\text{sync}}$  is the temporal anchor used to align the GUI visualization and the strumming playback logic.

The operational tempo shown in the interface is estimated locally around  $t_{\text{sync}}$ . When enough downbeats are available within a  $\pm 20$  s window, the backend computes the tempo from robust inter-downbeat intervals:

$$\widehat{\text{BPM}}_{\mathcal{D}} = \frac{60\hat{m}}{\text{mean}(\Delta\mathcal{D}_{\text{trim}})}, \quad (6)$$

where  $\Delta\mathcal{D}_{\text{trim}}$  denotes the local inter-downbeat intervals retained after IQR-based rejection and trimmed averaging. If

this estimate is unavailable, the backend uses local inter-beat intervals between 0.2 s and 2.0 s:

$$\widehat{\text{BPM}}_{\mathcal{B}} = \frac{60}{\text{mean}(\Delta\mathcal{B}_{\text{trim}})}, \quad (7)$$

where  $\Delta\mathcal{B}_{\text{trim}}$  is the corresponding robust set of local inter-beat intervals. A global median over valid beat intervals is used only as a final fallback.

Before being sent to the frontend, the synchronization results are converted into the unified data structure expected by the GUI. In this data structure, the beat sequence exposed as `beat_times` is limited to a window around  $t_{\text{sync}}$ , whereas the downbeat sequence is derived from the BeatNet downbeat detections. The AI-generated setup can therefore be formalized as:

$$\Theta = (\hat{c}, \widehat{\text{BPM}}, \mathcal{B}_{\text{GUI}}, \mathcal{D}, t_{\text{sync}}, \hat{m}), \quad (8)$$

where  $\hat{c}$  is the StrumKANet pattern suggestion,  $\widehat{\text{BPM}}$  is the operational practice tempo,  $\mathcal{B}_{\text{GUI}}$  is the beat sequence exposed to the browser,  $\mathcal{D}$  is the candidate downbeat set,  $t_{\text{sync}}$  is the synchronization anchor, and  $\hat{m}$  is the meter used by the GUI.

### C. Software Integration of the Neural Modules

In the current Strummin’ Gym prototype, the two neural modules are integrated as backend services. StrumKANet is called by the pattern inference wrapper, while BeatNet is called by the synchronization routine. The frontend does not execute neural inference directly; it receives normalized back-end outputs and exposes them as editable musical parameters in the Strumming Setup.

The deployed StrumKANet path is therefore restricted to the trained Rhythm Pattern (RP)-based configuration: a  $24 \times 60$  Rhythm Pattern tensor, a 2D convolutional encoder without attention, and a PatternKAN classification head. The BPM values returned by the same wrapper are auxiliary estimates computed outside the neural classifier and are used as metadata or fallback information rather than as StrumKANet predictions.

BeatNet is thus used only in the synchronization path. Its output is converted into the same frontend data structure used by the fallback synchronization routine. This allows the interface to retain the same frontend logic regardless of which synchronization path is used.

This integration strategy keeps neural inference server-side and musical interaction client-side. The backend computes the AI-derived suggestions and normalizes them into a stable data structure, while the browser uses the received values only to configure and schedule the interactive practice environment. Consequently, StrumKANet and BeatNet are not exposed as autonomous decision-making components: their outputs are presented as editable cues for pattern, BPM, meter, beat grid, downbeat grid, and synchronization point.

## V. RHYTHM-GESTURE WORKFLOW

The visible organization of the GUI is based on the distinction between Practice Mode and Song Mode. This distinction is meaningful because it separates two operational contexts.

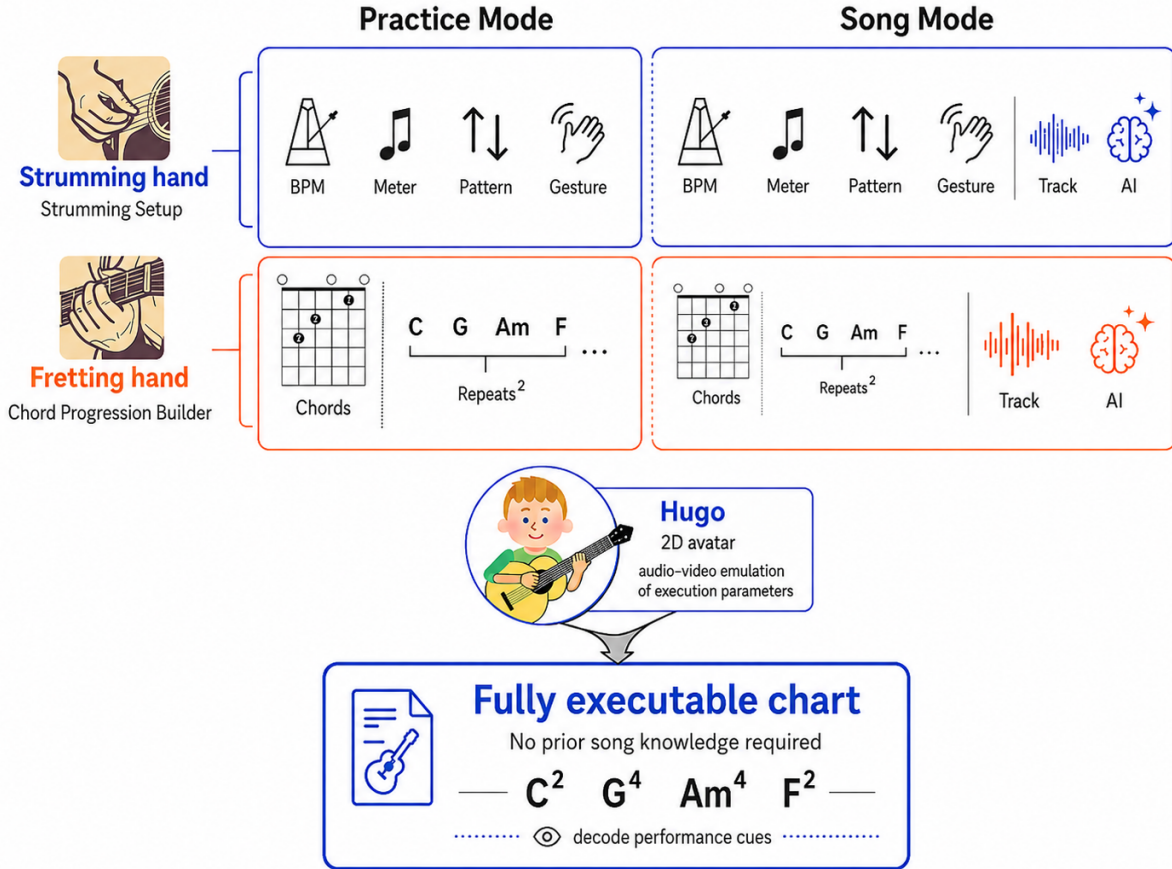


Fig. 3. Conceptual organization of Strummin' Gym as a fully executable chart workflow. Practice Mode and Song Mode define the two visible operational contexts of the GUI, while Strumming Setup and Chord Progression Builder constitute the deeper didactic-operational lines. Their integration makes pattern parameters explicit and links rhythmic structure to harmonic succession, producing a chord chart that can be performed without prior knowledge of the song.

Practice Mode allows the learner or teacher to build practice material without relying on a specific track, while Song Mode introduces a specific audio reference and possible AI-based suggestions for execution parameters. However, as shown in Fig. 3, both modes operate across the same deeper didactic structure: Strumming Setup and Chord Progression Builder.

**Strumming Setup.** The first didactic-operational line is dedicated to the strumming hand. Its purpose is to define the musical and gestural parameters that allow the learner to construct a constant down-up movement and internalize a rhythmic pattern as embodied action. The learner can manipulate BPM, meter, articulation, and pattern structure while observing and listening to the corresponding visual/auditory feedback. Hugo, the 2D strumming avatar implemented in the interface, supports this process by emulating the strumming gesture and the sonic result associated with each stroke (see Fig. 1). For a beginner, this is pedagogically relevant because the system externalizes a rhythm-gesture image that has not yet been internalized: the learner can see the down-up motion, hear the resulting articulation, and relate both to pulse, meter, subdivision, and pattern cyclicity.

In Practice Mode, Strumming Setup allows the learner or

teacher to isolate the gesture from a specific song and to work on foundational rhythmic materials.

In Song Mode, the same line is applied to a specific audio track: StrumKAN can provide strumming-pattern cues, while BeatNet can support beat, downbeat, meter, BPM, and synchronization information as reported in Fig. 4. The system therefore preserves the same gestural grammar across both visible modes.

The interface also includes a Custom Pattern Builder. This tool allows the learner or teacher to define a new pattern by selecting a three- or four-beat span and activating or deactivating individual down-up strokes within each beat. The resulting custom pattern is not a visual sketch only: it becomes part of the same execution logic used by the rest of the system, including beat-pattern rendering, Hugo's gesture visualization, chord-progression playback, and duration calculation. Thus, while the current StrumKAN-based recognition module operates on a limited set of learned pattern classes, the practice environment supports a substantially larger rhythmic vocabulary through manual custom-pattern construction.

**Chord Progression Builder.** The second didactic-operational line is dedicated to the fretting hand. Its role is

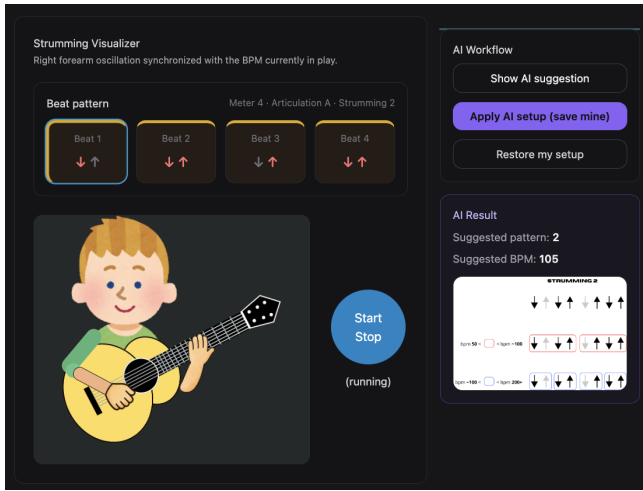


Fig. 4. Screenshot of Strummin' Gym browser interface for AI suggestions.

to connect chord positions and harmonic succession to the rhythmic pattern defined in Strumming Setup. The crucial parameter is the repeat value assigned to each chord. This value indicates how many complete cycles of the selected or user-created strumming pattern must be performed on that chord. If a chord is assigned two repeats, it lasts for two complete cycles of the pattern; if it is assigned four repeats, it lasts for four cycles. Since Strummin' Gym targets absolute beginners, the current chord vocabulary is intentionally limited to basic major and minor chords and simple seventh chords; higher chord extensions are outside the present pedagogical scope. Pattern-repetition values are displayed as superscripts, thus remaining typographically distinct from numerical chord-quality suffixes. This makes chord duration explicit without requiring conventional notation as shown in Fig. 5.

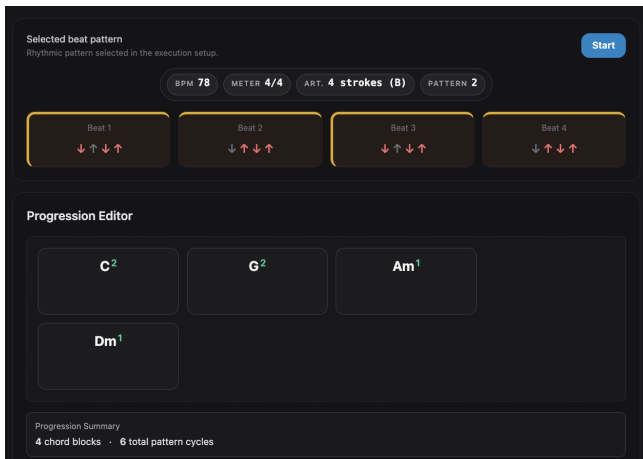


Fig. 5. Screenshot of Strummin' Gym browser interface for chord duration.

This repeat-based logic addresses a gap left by many chord-chart representations. In a conventional chart, even when a strumming pattern is provided, the learner may still need to know the song in advance in order to understand how long

each chord lasts. Strummin' Gym removes this dependency by encoding the relation between rhythmic structure and harmonic succession directly into the chart. A progression such as  $C^2 G^4 Am^4 F^2$  is therefore not only a harmonic sequence, but an executable rhythmic-harmonic instruction: each superscript indicates the number of pattern cycles assigned to the corresponding chord.

In Practice Mode, the Chord Progression Builder can be used to design harmonic exercises, chord-change studies, or teacher-built progressions. In Song Mode, it can be used to represent the harmonic structure of a specific song. In both cases, the same principle applies: the progression becomes performable because all execution-relevant parameters are explicit. The learner does not need to rely on memory of the original track; the chart itself contains the information needed to decode the performance.

This design supports teacher-led authoring directly. During a lesson, a teacher can define tempo, meter, articulation, chord order and repeat values, turning a class demonstration into an interactive and reproducible set of performance indications. In the current prototype, this does not yet correspond to a stored, shareable, or recallable workflow; rather, it describes the authoring logic already available during interaction.

Every parameter changes the enacted musical situation rather than a static label: BPM alters gesture density, articulation modifies the internal movement of the beat, meter reorganizes cyclic accentuation, and pattern choice determines which strokes sound and which remain silent gestures. The interface is therefore a constraint-based environment for non-linear practice. Hugo, active in both modes, externalizes this rhythm-gesture image for the beginner by rendering the down-up motion and its associated sound, mediating between auditory pulse, kinesthetic anticipation, and execution.

AI suggestions remain editable throughout: the user can display, apply, modify, or revert a suggestion, and after applying one, all execution parameters (i.e. BPM, meter, articulation, pattern, synchronization point, chord order, and repeats) stay modifiable.

This design is consistent with critical syntheses on AI-assisted music education that emphasize preserving teacher mediation and learner agency [8], [9].

Moreover, it is coherent with recent research showing that musicians often develop understanding through listening, imitation, and self-directed exploration [21].

## VI. DISCUSSION, LIMITATIONS, AND FUTURE WORK

**Discussion.** The prototype highlights a representational and pedagogical issue that is rarely made explicit in beginner-oriented guitar-learning systems: the gap between a song as an audio object, a chord chart as a symbolic object, and an accompaniment as an embodied action. For an absolute beginner, this gap cannot be addressed by merely adding missing information to the chart; such information must be embedded within a scaffolded rhythmic-gestural entry point that reduces cognitive load by limiting the initial decision space. Consistently with Bruner's view of learning as structured, revisitable

understanding [13], and with Perlak’s pedagogical reduction of beginner strumming to recurrent pop-rock patterns [17], Strummin’ Gym constrains the AI-assisted suggestion to one of three rhythmically and stylistically coherent patterns. This is especially relevant when the target song contains a technically advanced guitar part, or no explicit guitar strumming at all: the system does not reproduce the original arrangement, but proposes a playable accompaniment model that remains accessible to the beginner. The learner can then test either their own or the AI-suggested parameterization through Hugo’s audio-visual emulation, similarly to asking a teacher in presence to demonstrate and verify a possible execution. The editable practice configuration therefore functions as an intermediate representation between audio, chord symbols, and embodied execution.

This has consequences for how AI is used in the system. StrumKANet and BeatNet do not aim to produce a complete description of the song; rather, they initialize a configuration space in which the learner can listen, observe, compare, and revise. The relevant pedagogical question is not only whether the neural outputs are “correct”, but whether they are musically usable as starting points for practice. In this sense, the system shifts machine listening from classification to configuration: model outputs become valuable when tested against auditory perception, embodied execution, Hugo’s gesture emulation, and teacher-guided revision.

The integration of BeatNet also shows that synchronization is not a purely technical problem. A beat grid may be temporally stable while still being pedagogically problematic if the perceived first beat, the visual highlighting, and the strumming cycle do not coincide for the learner. For this reason, backend post-processing selects a synchronization anchor, derives a local operational BPM, and normalizes the result for browser-side use instead of exposing raw beat/downbeat detections. Phase, meter, and local tempo thus become part of the learning object: the learner is not only following a metronome, but working with an explicit model of where the pattern begins, how it cycles, and how it aligns with the audio track. The current synchronization logic does not explicitly detect the instrumental entrance: it selects the first BeatNet downbeat at or after 15 s, falling back to the first available beat, and to the full detected beat sequence when no post-15-s candidate is available. Consequently, long introductions are handled as long as BeatNet provides a reliable metrical grid, whereas non-metric or ambiguously tracked introductions may require a more adaptive anchor-selection strategy. A further implication concerns the difference between recognition vocabulary and practice vocabulary. The automatic StrumKANet suggestion is restricted to three learned pedagogical classes, but the practice environment is not. Through the Custom Pattern Builder, users can construct rhythmic configurations that exceed the classifier’s output space. The neural model therefore offers a constrained entry point into the song, while the interface preserves a broader authoring space for teachers and learners, avoiding the reduction of practice possibilities to what the model can recognize.

The Chord Progression Builder extends this representational logic to harmony. Its main contribution is not the manual insertion of chords, but the use of the pattern cycle as a durational unit. In conventional chord charts, duration is often inferred from prior listening or stylistic knowledge; in Strummin’ Gym, duration is encoded as a number of pattern repetitions. This creates a meso-level representation situated between standard notation and informal chord symbols: more explicit than a chord sheet, but less demanding than full rhythmic notation. For beginners, this is pedagogically relevant because it connects harmonic change to a bodily repeatable gesture.

Finally, the absence of automatic performance grading is not merely a missing feature, but a design stance. Since the prototype does not require sensing the learner’s actual performance, it can provide an embodied model of the gesture without turning practice into continuous surveillance or score-based evaluation. Hugo does not judge the learner; he externalizes a rhythm-gesture image that can be imitated, tested, discussed, and modified. This supports a form of AI-assisted music education in which the system structures the conditions for musical understanding while leaving interpretive authority with the learner and the teacher.

**Limitations.** The current prototype has five main limitations. First, automatic strumming-pattern recognition is restricted to the three pedagogical classes learned by StrumKANet. The Custom Pattern Builder extends the practice vocabulary, but not the automatic suggestion space. Second, the AI-assisted workflow currently covers rhythmic-pattern classification and beat/downbeat synchronization, but not harmonic analysis: chord progressions and chord-duration values are still entered manually. Third, the prototype supports interactive configuration and exercise construction, but does not yet implement persistent workflow storage, workflow recall, teacher–student sharing, or high-quality printable export. Fourth, online deployment requires further work on containerization, scalability, storage policy, dependency management, and rights-cleared repertoire management. Network-related latency and scalability will become relevant in a future remotely deployed version and therefore require dedicated profiling and evaluation. Fifth, the pedagogical effectiveness of the system has not yet been evaluated through structured studies with teachers and beginner learners. Accordingly, the present work should be interpreted as a system-design and functional-integration study rather than as evidence of pedagogical effectiveness.

**Future work** will focus on four main directions. First, the AI-assisted workflow will be extended from rhythmic pattern recognition and beat/downbeat synchronization toward harmonic support, potentially integrating Automatic Chord Recognition methods to propose chord progressions and chord-duration suggestions from a reference track while keeping them fully editable by the user. Second, the prototype will be evaluated with teachers and beginner guitarists in structured pedagogical settings, with attention to usability, rhythmic-gestural awareness, and the effectiveness of editable

AI suggestions compared with conventional chord-chart practice. Third, the current interactive configuration will be developed into a persistent learning workflow through a database-backed extension of the Data Layer. This includes saving and recalling complete practice setups, sharing them between teacher and learner, and generating printable or digital learning materials that preserve BPM, meter, articulation, pattern representation, synchronization information, and chord progression with pattern repetitions. Fourth, future online versions will require further work on deployment robustness, rights-cleared audio sourcing, data minimization, consent, storage policy, and transparency of AI-generated suggestions, following ethical research on IoMusT and educational AI [15], [16].

## VII. CONCLUSIONS

This paper presented Strummin' Gym, a web-based environment for beginner pop-rock guitar learning framed within music education applications for the Internet of Musical Things. The system addresses a pedagogical gap left by common lyrics-and-chords resources: while these materials usually provide harmonic symbols, they rarely make explicit the rhythmic, gestural, and durational information required to perform a strummed accompaniment. Strummin' Gym responds to this limitation by transforming audio-informed rhythmic analysis into editable multimodal practice configurations.

The current prototype is organized around two complementary learning contexts. Practice Mode supports the construction of strumming exercises without requiring a specific audio track, while Song Mode connects the same pedagogical logic to a target song and to AI-assisted rhythmic suggestions. Within the Song Mode Strumming Setup, StrumKANet suggests one of three pedagogically defined strumming patterns, while BeatNet provides BPM estimation and a beat/downbeat reference grid for synchronization. In the implemented pipeline, StrumKANet is used as a categorical pattern classifier based on RP\_extract rhythm-pattern features, a 2D convolutional encoder without attention, and a PatternKAN classification head. BeatNet is integrated as the synchronization backend through its pretrained CRNN model with offline DBN inference. Their outputs are not treated as final automatic annotations, but as editable cues that the learner or teacher can listen to, validate, correct, and customize before practice.

A second stage, the Chord Progression Builder, extends the rhythmic configuration defined in the Song Mode Strumming Setup to harmonic practice. By assigning each chord a duration expressed as repetitions of the selected or user-created strumming pattern, the system converts a conventional chord sequence into an executable rhythmic-harmonic object. This makes explicit the relation between pulse, gesture, chord change, and pattern repetition, allowing the performer to study an accompaniment through encoded performance cues rather than through prior familiarity with the song.

The contribution of Strummin' Gym is therefore not automatic performance assessment, but the design of a pedagogically grounded IoMusT workflow in which neural rhythm anal-

ysis supports embodied musical understanding. The separation between server-side MIR/neural processing and client-side browser interaction enables the system to combine audio analysis, metronome synchronization, beat-pattern visualization, and Hugo's audio-visual gesture emulation within a responsive practice interface. Neural outputs remain transparent, editable, and subordinated to musical practice, positioning Strummin' Gym as a tool for scaffolding rhythmic-gestural awareness in early pop-rock guitar accompaniment.

## REFERENCES

- [1] M. G.-U. G. d. León, "Guitar teaching: State of the art and research questions," *Society. Integration. Education. Proceedings of the International Scientific Conference*, vol. 4, pp. 361–367, May 2019. [Online]. Available: <https://journals.rta.lv/index.php/SIE/article/view/3952>
- [2] R. C. Rodriguez and V. Marone, "Guitar learning, pedagogy, and technology: A historical outline," *Social Sciences and Education Research Review*, vol. 8, no. 2, pp. 9–27, Dec. 2021. [Online]. Available: <https://zenodo.org/records/5784696>
- [3] "Ultimate Guitar | Muse Group." [Online]. Available: <https://www.mu.se/ultimate-guitar>
- [4] "Ultimate Guitar: Chords & Tabs App - App Store." [Online]. Available: <https://apps.apple.com/by/app/ultimate-guitar-chords-tabs/id357828853>
- [5] M. Pennese, S. Giacomelli, and C. Rinaldi, "A Kolmogorov Arnold Network NAS Framework for Strumming Pattern Recognition in Technology-Enhanced Pop/Rock Music Education," in *2025 IEEE 6th International Symposium on the Internet of Sounds (IS2)*. IEEE, 2025, pp. 1–10. [Online]. Available: <https://ieeexplore.ieee.org/abstract/document/11284580/>
- [6] L. Nijs and L. Turchet, "Internet of Musical Things for Children," in *2024 IEEE 5th International Symposium on the Internet of Sounds (IS2)*. Erlangen, Germany: IEEE, Sep. 2024, pp. 1–9. [Online]. Available: <https://ieeexplore.ieee.org/document/10704214/>
- [7] S. Gallagher, *Embodied and Enactive Approaches to Cognition*, 1st ed. Cambridge University Press, Oct. 2023. [Online]. Available: <https://www.cambridge.org/core/product/identifier/9781009209793/type/element>
- [8] C. A. N. Mazlan, H. F. Hanafi, M. R. Sarifin, A. R. Md Noor, S. A. Sadykova, R. Hidayatullah, and S. Jamnongsarn, "Artificial intelligence applications and pedagogical challenges in music education," *Discover Education*, vol. 5, no. 1, p. 140, Jan. 2026. [Online]. Available: <https://link.springer.com/10.1007/s44217-026-01127-3>
- [9] J. F. Merchán Sánchez-Jara, S. González Gutiérrez, J. Cruz Rodríguez, and B. Syroyid Syroyid, "Artificial Intelligence-Assisted Music Education: A Critical Synthesis of Challenges and Opportunities," *Education Sciences*, vol. 14, no. 11, p. 1171, Oct. 2024. [Online]. Available: <https://www.mdpi.com/2227-7102/14/11/1171>
- [10] C. Greco, L. A. Ludovico, and P. C. Rivoltella, "Technology and music: The role of AI in instrumental and compositional education," *EDULEARN25 Proceedings*, pp. 8707–8714, 2025, conference Name: 17th International Conference on Education and New Learning Technologies ISBN: 9788409742189 Meeting Name: 17th International Conference on Education and New Learning Technologies. [Online]. Available: <https://library.iated.org/view/GRECO2025TEC>
- [11] V. L. Demenescu and I. A. Ardeanu, "Didactics of Music Theory through Ear Master - Case Study," *ICT in Musical Field / Tehnologii Informatice si de Comunicatie in Domeniul Muzical*, vol. 16, no. 1, p. 51, Jan. 2025. [Online]. Available: <https://openurl.ebsco.com/contentitem/gcd:186239166?sid=ebsco:plink:crawler&id=ebsco:gcd:186239166>
- [12] A. M. Freschi and R. Neulichedl, *Metodologia dell'insegnamento strumentale: aspetti generali e modalità operative*. Pisa: Edizioni ETS, 2012, oCLC: 955947024.
- [13] J. S. Bruner, *The Process of Education, Revised Edition*. Harvard University Press, Jun. 2009, google-Books-ID: S6FKW90QY40C.
- [14] H. Honing, "Without it no music: beat induction as a fundamental musical trait," *Annals of the New York Academy of Sciences*, 2012.
- [15] J. Brusseau and L. Turchet, "An Ethics Framework for the Internet of Musical Things," *IEEE Transactions on Technology and Society*, pp. 1–1, 2024, conference Name: IEEE Transactions on Technology

- and Society. [Online]. Available: <https://ieeexplore.ieee.org/document/10538251/?arnumber=10538251>
- [16] Directorate-General for Education, Youth, Sport and Culture (European Commission), *Guidelines on the ethical use of artificial intelligence and data in teaching and learning for educators*. Publications Office of the European Union, 2026. [Online]. Available: <https://data.europa.eu/doi/10.2766/7967834>
- [17] K. Perlak, *Berklee Beginning Guitar*. Berklee Press, 2024.
- [18] M. Heydari, F. Cwitkowitz, and Z. Duan, “BeatNet: CRNN and Particle Filtering for Online Joint Beat Downbeat and Meter Tracking,” Aug. 2021, arXiv:2108.03576 [eess.AS]. [Online]. Available: <http://arxiv.org/abs/2108.03576>
- [19] T. Lidy and A. Rauber, “Evaluation of Feature Extractors and Psycho-acoustic Transformations for Music Genre Classification,” in *Proceedings of the Sixth International Conference on Music Information Retrieval (ISMIR 2005)*, London, UK, September 11-15 2005, pp. 34–41.
- [20] Z. Bozorgasl and H. Chen, “Wav-KAN: Wavelet Kolmogorov-Arnold Networks,” *arXiv preprint arXiv:2405.12832*, 2024. [Online]. Available: <https://arxiv.org/abs/2405.12832>
- [21] R. Wright and P. Kanellopoulos, “Informal music learning, improvisation and teacher education,” *British Journal of Music Education*, vol. 27, no. 1, pp. 71–87, Mar. 2010. [Online]. Available: [https://www.cambridge.org/core/product/identifier/S0265051709990210/type/journal\\_article](https://www.cambridge.org/core/product/identifier/S0265051709990210/type/journal_article)