COGNITIVELY-AWARE SYMBOLIC NOTATION VIA SIMULATION: A SOFTWARE-ONLY FRAMEWORK FOR ADAPTIVE MUSIC REPRESENTATION

Sanjay Majumder
Style tree
sanjay@styletree.me

Neal Anderson
Style tree
neal@styletree.me

ABSTRACT

This paper presents a software-only, simulation-based framework for cognitively-aware music notation. The system provides a modular proof-of-concept for adaptive music representation that responds to modeled performer cognition without requiring hardware-based brain–computer interfaces. Instead of using physiological signals, the framework employs behavioral proxies, including performance timing, expressive deviation, and simulated eye-gaze, to infer cognitive load and attentional focus. These simulated cues drive real-time adjustments to the notation, such as adaptive visual spacing, emphasis modulation, and rhythmic simplification, aligning score presentation with predicted attentional bandwidth. The framework integrates symbolic music processing and machine-learning heuristics within a unified architecture composed of a Symbolic Music Engine, Cognitive-State Simulator, and Adaptive Notation Engine. Evaluation was conducted entirely in a closed-loop simulation environment using professional score datasets and virtual performer profiles of varying expertise. The results demonstrate improved simulated sight-reading fluency, reduced modeled cognitive effort, and improved readability. In general, this study establishes the feasibility of purely software-driven adaptive notation, offering a foundation for future empirical validation and interdisciplinary research connecting computational notation design, cognitive modeling, and AI-assisted music performance.

1. INTRODUCTION

Music notation has historically served as the primary interface between composer intent and performer interpretation. From neumes in medieval chant to modern staff notation and experimental graphic scores, the evolution of notational systems has mirrored broader technological and cultural shifts. In recent decades, the integration of computational methods has significantly expanded the representational capacity of music notation, enabling real-time rendering, interactive performance systems, and algorithmic composition. However, traditional notation systems often remain static and do not account for the dynamic cognitive states of performers, particularly under real-time constraints such as sight-reading, improvisation, or multitasking.

This research addresses the growing need for adaptive and cognition-aware music notation systems developed purely through software. Unlike brain-computer interfaces (BCIs) that rely on external neural sensors, our approach leverages simulated cognitive signals derived from software-based performance analysis, such as expressive timing, cursor tracking, and modeled attentional dynamics. These behavioral signals act as proxies for the underlying mental states and enable the adaptive rendering of symbolic music in real-time.

Drawing upon advances in artificial intelligence, symbolic music modeling, and cognitive science, this work introduces a modular software framework that simulates neural-informed adaptation of music scores. The system is designed to enhance the engagement of the performer, reduce the cognitive load, and promote fluid interpretation by continuously reshaping the notation based on the inferred mental workload. This approach aligns with the central theme of Notational Intelligence by treating notation not merely as a representational artifact but as a responsive medium grounded in real-time human-computer interaction.

This work presents a computational framework rather than a performer-in-the-loop system. We outline the design, implementation, and evaluation of a software-only architecture that augments notational intelligence without requiring hardware-based neural interfaces. Through simulation, we examine the effects of adaptive notation on fluency, error rates, and subjective experience, offering new insights at the intersection of music representation, cognitive modeling, and computational design. The current implementation is limited to simulation-based evaluation, serving as a proof of concept and establishing a theoretical and technical foundation for future empirical validation with live musicians.

2. BACKGROUND AND RELATED WORK

Music notation serves as a primary interface between a composer's intent and a performer's interpretation, translating abstract musical ideas into structured symbolic representations. Traditional Western notation, while deeply entrenched and highly expressive, is inherently

Copyright: ©2025 Sanjay Majumder and Neal Anderson. This is an open-access article distributed under the terms of the Creative Commons Attribution 3.0 Unported License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

static—providing the same representation regardless of the performer's skill level, cognitive state, or performance context. Although this has allowed the standardization and preservation of musical works, it can create barriers for novice and intermediate performers who may struggle with real-time interpretation.

Research has shown that comprehension of notation systems is shaped by cognitive factors such as working memory, visual attention, and perceptual grouping [1, 2]. For example, Cheng et al. [1] systematically evaluated ten different notation systems and demonstrated that design variations significantly influence readability and cognitive load. Neurocognitive studies further reveal that music reading engages intertwined visual–motor and auditory–cognitive networks that overlap with language and spatial processing systems [3, 4]. Proverbio and Sanoubari [3] found that music literacy modifies hemispheric specialization in word reading areas, evidencing strong neural overlap between linguistic and musical processing. Similarly, Cara [4] observed that sight-reading experience can enhance spatial working memory and metacognitive awareness, suggesting cross-domain transfer effects.

The cognitive demands of traditional notation have motivated the exploration of adaptive systems that can dynamically align visual representation with the perceptual and attentional capacities of the performer. Recent neuroscientific reviews emphasize that musical cognition involves predictive coding, neural synchronization, and embodied action, all of which could inform adaptive notation that responds to real-time performer states [5, 6]. De Souza et al. [5] argue that such mechanisms act as cognitive scaffolds, modulating attention and memory during performance.

Prior work in computer-assisted music performance has focused primarily on score-following and accompaniment systems. For example, Raphael's initial framework for score following [7] advanced the alignment between live audio and symbolic representation but did not modify the notation in response to the state of the performer. More recent tools such as Hyperscore [8] and INScore [9] introduced dynamic graphical representations and interactive composition environments, yet they target compositional visualization rather than real-time performance adaptation.

Several studies have also examined how notation design and layout affect sight-reading and performance accuracy. Stenberg and Cross [10] demonstrated that increased spatial whitespace in scores improves reading fluency by reducing visual crowding. Fürst et al. [11] proposed "Rhythmic Fingerprints," which overlay rhythmic density cues to aid interpretation of complex passages. Parallel neurocognitive findings of Wong et al. [2] indicate that visual training with musical symbols alters cortical responses, reinforcing the importance of notational visual design.

From a computational perspective, adaptive visualization and music information retrieval (MIR) systems have sought to synchronize notation with tempo, gaze, or performance dynamics. However, most lack integration of cognitive modeling or neurofeedback. Emerging research on cognitive load estimation and neural signal decoding [6, 5] opens pathways toward real-time adaptive notation—where visual complexity and density adjust dynamically to performer attention, stress, or fluency levels.

Despite these advances, a comprehensive framework that unifies symbolic music processing, neural signal interpretation, and real-time adaptive rendering remains underexplored. Our work addresses this gap by developing a system that integrates cognitive feedback into the notation rendering process, extending prior visual and computational research toward a context-aware, performer-responsive paradigm for adaptive notation in live music performance.

3. SYSTEM ARCHITECTURE

The proposed software framework consists of six interdependent core modules that simulate cognitively-informed music notation without requiring physical neural interfaces. Each module is implemented as a standalone component with clearly defined input-output interfaces, supporting modular experimentation and refinement. The architecture is fully software-based and operates in a real-time symbolic music environment, processing MIDI scores, performer actions, and simulated cognitive cues to dynamically modify both the visual and semantic representation of music.

Figure 1 illustrates the high-level architecture and module hierarchy. The core modules, Symbolic Music Engine (SME), Neural Signal Interface (NSI), Cognitive-State Analyzer (CSA), Adaptive Notation Renderer (ANR), Feedback Loop and Latency Management, and Logging & Data Export, form the foundational pipeline. Four extended modules augment this architecture for simulation purposes: Symbolic Score Processor (SSP) elaborates the SME by providing symbolic transformations; Cognitive State Simulator (CSS) substitutes the CSA when real cognitive data are unavailable; Adaptive Notation Engine (ANE) expands the ANR into a fully adaptive module; and Performance Simulator (PS) models performer behavior downstream. Dashed arrows in Figure 1 indicate extensions or substitutions of core functionality by these modules.

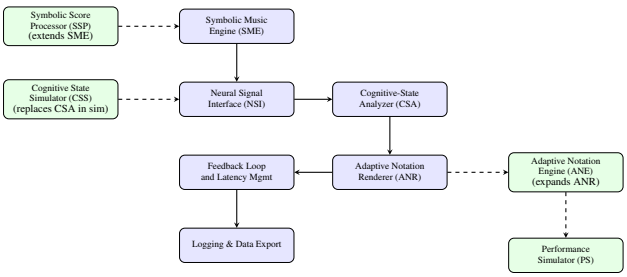

Figure 1. System architecture hierarchy. Core modules (blue) form the base framework, while extended modules (green) expand or substitute functionality for simulation-based evaluation.

3.1 Symbolic Music Engine (SME)

The SME is responsible for parsing, storing, and manipulating musical scores in symbolic form (e.g., MusicXML or MIDI). It supports both standard notation rendering and internal transformations required by downstream modules. The engine includes a rule-based harmonic and rhythmic analysis layer that maintains semantic integrity during adaptation, allowing contextual modifications such as voice rebalancing or rhythmic simplification.

3.2 Neural Signal Interface (NSI)

Although we retain the historical name Neural-Signal Interface (NSI), the current prototype does not read physiological signals. Instead of hardware-based neural data, the NSI synthesizes proxies for cognitive and attentional states using observable performance features. These proxies include real-time tempo deviation, articulation variance, pitch accuracy, and simulated eye-tracking (based on scrolling or pointer focus). Machine learning models trained on annotated performance datasets map these behavioral inputs to cognitive dimensions such as mental load, attentional drift, and working memory stress.

3.3 Cognitive-State Analyzer (CSA)

The CSA interprets the outputs of the NSI to maintain a continuously updated profile of the cognitive state of the performer. A temporal smoothing algorithm (e.g., Kalman filtering) is applied to prevent erratic updates, and a hierarchical classifier maps this profile to predefined notational strategies (e.g., simplify rhythmic groupings, highlight voice-leading). The CSA acts as the core of the decision-making system.

3.4 Adaptive Notation Engine (ANE)

The Adaptive Notation Engine (ANE) dynamically modifies the visual presentation of the score based on the instructions from the CSA. Changes include spacing adjustments, reduction of polyphonic density, adaptive coloration, and annotation overlays. All changes are reversible and non-destructive, preserving original musical semantics while optimizing readability. Rendering is managed via a web-based canvas engine with SVG-based symbol manipulation.

3.5 Feedback Loop and Latency Management

A software-level feedback loop synchronizes performance inputs with rendering changes while maintaining sub-50ms latency to preserve responsiveness. The loop includes predictive modeling for proactive adjustments (e.g., anticipatory highlighting based on gaze simulation) and rollback mechanisms to avoid visual instability. All computations are optimized for minimal overhead within standard computing environments.

3.6 Logging and Data Export Module

This module records performance metrics, inferred cognitive states, and notation adaptation events in real-time. The data is exportable in CSV and JSON formats for post-hoc analysis. A structured logging scheme allows for the evaluation of how cognitive state transitions correlate with visual modifications and musical performance outcomes.

The architecture is implemented entirely in Python and JavaScript using open-source libraries such as `music21` [12], `TensorFlow`, and `VexFlow`. The system is designed for extensibility, supporting alternative models of cognition or notation, and integrates seamlessly with DAWs and digital notation software via MIDI or OSC protocols. This modular, purely software-based system serves as a foundation for future developments in notational intelligence grounded in cognitive modeling.

4. SYMBOLIC SCORE PROCESSOR (SSP)

The **Symbolic Score Processor (SSP)** is the foundational module of the system, responsible for parsing, interpreting, and transforming symbolic music data into a computationally tractable and cognitively adaptive representation. It is implemented as a modular software stack built on top of symbolic music processing libraries, such as `music21` [12], enabling compatibility with common score formats, including MusicXML, MIDI, and MEI.

4.1 Architecture and Data Representation

Upon ingestion of a symbolic file, the SSP constructs a multi-layered internal representation:

- **Note Event Graph (NEG)**: A directed graph $G = (V, E)$ where V denotes pitch-time-duration triplets and E encodes temporal and voice-continuity relationships.
- **Hierarchical Score Tree (HST)**: A tree-based structure encoding nested metrical levels (beat $\rightarrow$ measure $\rightarrow$ phrase $\rightarrow$ section), facilitating transformations at different granularities.
- **Semantic Annotation Matrix (SAM)**: A sparse matrix overlay where each element stores tag-based metadata for events (e.g., harmonic function, cadential strength, articulation complexity).

4.2 Functional Modules

The SSP pipeline includes the following core modules:

1. **Semantic Parser**: Identifies and labels harmonic progressions, voice leading, phrase boundaries, and syncopation patterns using rule-based and trained pattern-matching algorithms.
2. **Event Abstraction Engine**: Groups atomic note events into musically significant constructs such as motivic cells, tuplets, or chord aggregates. This abstraction supports high-level structural manipulations rather than low-level event editing.
3. **Temporal Alignment Layer**: Maps symbolic events to both absolute timestamps and symbolic locations (e.g., beat 2.5 of measure 4), enabling

bidirectional synchronization with performance data streams and simulated attentional events.

4. **Transform Engine**: Enables dynamic, rule-based or ML-driven score manipulations, including:
    - *Density reduction*: Collapsing polyphonic layers or removing non-structural notes.
    - *Complexity adaptation*: Simplifying rhythmic subdivisions based on predicted cognitive load.
    - *Contour emphasis*: Enhancing melodic or harmonic skeletons to foreground essential musical information.
5. **Metadata Annotator**: Annotates symbolic material with difficulty metrics (e.g., intervallic entropy, rhythmic irregularity), attention-critical zones (e.g., rapid changes, large leaps), and stylistic priors (e.g., idiomatic usage patterns).

4.3 Integration with the Cognitive Pipeline

The SSP is designed to interface seamlessly with upstream and downstream modules:

- Inputs from the **Cognitive-State Analyzer (CSA)** are used to trigger symbolic transformations in real-time, such as increasing visual spacing during high cognitive load or reducing note density during attentional dips.
- Outputs from the SSP feed into the **Adaptive Notation Renderer (ANR)** to generate notation views tailored to current user state models.

4.4 Software Properties

The SSP supports real-time symbolic score adaptation under the following constraints:

- **Latency budget:** $<$20 ms per adaptation cycle.
- **Complexity scaling:** $O(n \log n)$ symbolic token processing with caching mechanisms for harmonic and rhythmic lookups.

By combining structural musical knowledge with flexible symbolic transformations, the SSP acts as a cognitively-aware symbolic interpreter, ensuring that any adaptation of the musical score remains musically coherent and perceptually optimized.

5. COGNITIVE STATE SIMULATOR (CSS)

The **Cognitive State Simulator (CSS)** models the dynamic attentional and perceptual state of a notional performer or user interacting with a symbolic score. It serves as a software-only proxy for neurophysiological inputs, enabling system evaluation and user modeling without requiring real-time neural data. By simulating cognition-driven parameters such as attention span, working memory load, and perceptual salience, CSS bridges symbolic music content with plausible cognitive responses.

5.1 Simulation Model

The CSS is implemented as a hybrid stochastic-cognitive model inspired by established theories in music cognition and attention. The model operates on a multi-timescale framework and comprises the following modules:

- **Attentional Focus Model (AFM)**: Simulates a moving window of attentional bandwidth, constrained by tempo, note density, and musical complexity. The attentional window W_t is updated at each timestep using:

$$W_{t+1} = W_t + \Delta W_t \tag{1}$$

where

$$\Delta W_t = \alpha \cdot (C_t - \bar{C}) + \beta \cdot \text{ISI}_t + \gamma \cdot \text{Entropy}_t \tag{2}$$

where all terms are normalized to $[-1, 1]$. C_t is instantaneous symbolic complexity, $\bar{C}$ its running mean, ISI_t is the inter-onset interval deviation, and Entropy_t is the local information entropy of recent note events. The parameters α, β, and γ were tuned empirically (0.3, 0.2, 0.5). W_t denotes the number of events currently within the attentional window.

- **Working Memory Buffer (WMB)**: Tracks recent melodic, harmonic, and rhythmic motifs to simulate capacity-limited retention. Implements a decaying memory trace model:

$$M(t) = M_0 \cdot e^{-\lambda t} + \kappa \cdot \text{Rehearsal}_t \tag{3}$$

Here $M(t)$ models active symbolic memory. The parameter λ controls decay ($0.15\,\text{s}^{-1}$) and κ scales rehearsal reinforcement. The term $\text{Rehearsal}_t \in [0, 1]$ represents rehearsal intensity computed from recent repetition rate. Time t is measured in seconds.

This informs score transformations by identifying overloading sections that exceed a theoretical working memory threshold.

The AFM is inspired by the Kahneman's capacity model of attention [13] and large literature on temporal expectancy in music cognition, while the WMB follows a decay and rehearsal structure consistent with Baddeley's working memory model [14, 15]. This grounding ensures that simulated attentional and memory dynamics reflect established cognitive theories, though at an abstracted level.

- **Surprise and Salience Estimator (SSE)**: Computes the expected information content (surprise) of incoming events using a Markov or transformer-based n-gram statistical model trained on symbolic corpora. Events with high salience scores (e.g., large pitch leaps, syncopations) are marked for possible emphasis or reduction based on the user profile.
- **User Profile Parameters**: Adjustable parameters include:

- *Expertise Level*: modulates the tolerance to complexity.
- *Cognitive Load Sensitivity*: adjusts how quickly overload is triggered.
- *Familiarity*: affects pattern recognition and motif expectancy.

5.2 Output Parameters

The CSS outputs a time-aligned stream of cognitive-state vectors $S_t = [a_t, m_t, s_t]$ where:

- a_t: instantaneous attention score (range 0 to 1),
- m_t: memory buffer saturation (0 to 1),
- s_t: salience index of the current musical event.

These outputs are sent in real-time to the **Symbolic Score Processor (SSP)** and the **Adaptive Notation Renderer (ANR)**, where they inform layout adjustments, visual emphasis, and content pruning strategies.

5.3 Evaluation Modes

The CSS supports three simulation modes:

1. **Preset Mode**: Uses predefined trajectories of attention and memory to benchmark system behavior under known constraints.
2. **Score-Coupled Mode**: Dynamically derives state transitions from the symbolic input score using complexity-aware heuristics.
3. **Corpus-Learned Mode**: Employs machine learning models trained on human annotation or EEG-informed datasets to predict plausible state trajectories.

5.4 Software Implementation

- Implemented in `Python 3.11` using `NumPy`, `SciPy`, and `PyTorch` for simulation and modeling.
- All cognitive traces are logged as JSON streams, compatible with downstream visualization and debugging tools.
- Modular interface allows plug-in replacement with real-time neuroadaptive models when EEG or fNIRS inputs are available in future expansions.

The CSS thus acts as a cognitive surrogate that enables adaptive notation systems to be developed, tested, and evaluated without the logistical overhead of physiological sensing, while remaining grounded in scientifically validated models of musical cognition.

6. ADAPTIVE NOTATION ENGINE (ANE)

The **Adaptive Notation Engine (ANE)** serves as the central module responsible for generating context-aware, cognitively responsive music notation. It dynamically modifies the graphical layout, content density, and symbolic emphasis of a musical score based on real-time cognitive state inputs and user preferences. The ANE enables the rendering of notation that adapts not only to musical structure but also to user-specific perceptual and cognitive capabilities.

6.1 Core Objectives

The ANE aims to:

- Reduce cognitive overload during score reading by simplifying or suppressing low-priority content.
- Enhance readability through layout optimization, spacing, and symbol selection.
- Maintain musical integrity and semantics while adapting the presentation.
- Support diverse user profiles (e.g., beginner vs. expert, low vs. high attention).

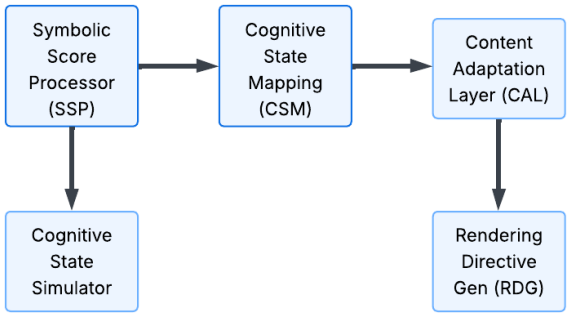

Figure 2. Pipeline of the Adaptive Notation Engine (ANE).

Here, "musical integrity" refers to preserving the underlying harmonic and rhythmic skeleton of a passage even when ornaments or polyphonic density are reduced. "Semantic preservation" means that transformations never alter structural functions such as cadence points, harmonic progressions, or motivic contours, ensuring that the score remains musically faithful while visually simplified.

6.2 Input Interface

The ANE accepts a rich set of inputs:

- Symbolic Score Stream from the **Symbolic Score Processor (SSP)**.
- Cognitive State Vectors from the **Cognitive State Simulator (CSS)** or **Cognitive-State Analyzer (CSA)**.
- User Profile Metadata (expertise level, display preferences, prior exposure).

- Performance Context Flags (e.g., live vs. rehearsal mode).

6.3 Transformation Pipeline

The ANE employs a multi-stage transformation pipeline, consisting of the following components:

6.3.1 Complexity Assessment Unit (CAU)

Analyzes incoming score passages to quantify harmonic density, rhythmic irregularity, pitch range, polyphonic texture, and ornamentation. Computes a *complexity score* C_s for each segment using a weighted formula:

$$C_s = w_p P + w_r R + w_h H + w_o O \tag{4}$$

where P, R, H, O represent pitch variability, rhythmic deviation, harmonic density, and ornamentation level respectively. All variables are normalized to $[0, 1]$ for comparability. The weights are defined as $w_p = 0.25$, $w_r = 0.25$, $w_h = 0.3$, and $w_o = 0.2$.

Complexity and familiarity functions are computed independently from the Cognitive State Simulator to avoid circularity in evaluation.

6.3.2 Cognitive State Mapping (CSM)

Maps the cognitive state vector $S_t = [a_t, m_t, s_t]$ to adaptive parameters such as symbol suppression thresholds, detail scaling factors, and attention highlight weights. For instance:

$$D_t = f(a_t, m_t) = \delta \cdot (1 - a_t) + \mu \cdot m_t \tag{5}$$

where D_t is a display simplification factor, and δ, μ are tunable weights.

6.3.3 Content Adaptation Layer (CAL)

Applies cognitive-state-aware transformations such as:

- *Note Reduction*: Omits grace notes, inner voice notes, or repetitive patterns.
- *Simplified Notation*: Converts tuplets to even rhythms, reduces polyphonic texture.
- *Symbol Emphasis*: Enlarges or colors structurally salient events (e.g., syncopations, cadences).
- *Temporal Stretching*: Allocates more visual space to rhythmically dense segments.

6.3.4 Rendering Directive Generator (RDG)

Converts adapted symbolic instructions into rendering directives compliant with rendering backends such as `Verovio` [16]. It supports MusicXML and MEI export with embedded adaptive metadata.

6.4 Dynamic Layout Policies

Layout adaptation policies are designed based on the following:

- **Spatial Budget**: Enforces maximum notes-per-line and minimum spacing thresholds.
- **Visual Salience Map**: Prioritizes spacing and visual emphasis according to current attentional state.
- **Continuity Constraints**: Avoids abrupt graphical changes that may disorient the user (e.g., keep layout stable over multiple bars unless a strong cognitive state transition occurs).

6.5 Technical Implementation

- Written in `Python 3.11` with `music21` [12] for symbolic processing and `Verovio` for MEI rendering.
- Uses `PyTorch` models to interpret cognitive-state embeddings into symbolic operations.
- Modular architecture allows backend substitution (e.g., real-time web rendering with VexFlow).
- All transformations are logged in a reversible, traceable format to allow playback or debugging.

6.6 Example Output

Given a passage with high complexity and low attention score ($a_t < 0.4$), the ANE may:

- Remove ornamentations and arpeggios.
- Convert 7-tuplets into binary subdivisions.
- Color-code key transition points in red.
- Add visual spacing between syncopated rhythms.

These modifications are rendered in real-time and are updated dynamically as new cognitive state vectors arrive. This enables a fluid, responsive notation experience that aligns with the user's real or simulated cognitive engagement during performance or study.

7. PERFORMANCE SIMULATOR (PS)

The **Performance Simulator (PS)** models a dynamic virtual performance environment, with no human performer involvement, designed to emulate musician behavior under cognitively adaptive notational conditions. It serves as a software-only testing ground for iteratively evaluating the effectiveness of adaptive notation in real-time and historical playback contexts.

7.1 Architecture

The PS operates as a modular simulation engine with the following core components:

- **Temporal Engine (TE)**: Manages tempo, rhythmic precision, and expressive timing with adjustable performance profiles (e.g., legato, rubato, swing).
- **Gesture Interpreter (GI)**: Translates symbolic score events into simulated motor gestures (keystrokes, bowing directions) using a gesture grammar defined by the performance context.
- **Cognitive Lag Model (CLM)**: Introduces timing deviations and visual processing delays based on complexity, familiarity, and attentional state.
- **Error Injector (EI)**: Simulates memory slips, hesitations, or finger errors to mimic human inaccuracies, guided by empirical music performance data.

7.2 Technical Workflow

The PS is built as a discrete-event simulation engine. The symbolic score S and the cognitive state vector S_t are used to modulate performance dynamics and execution fidelity. Each note event $n_i \in S$ is processed as follows:

7.2.1 Latency Calculation

$$\begin{aligned}\Delta_i = \alpha \cdot \text{Complexity}(n_i) \\ + \beta \cdot (1 - \text{Familiarity}(n_i))\end{aligned} \tag{6}$$

where α and β are weights determined through empirical calibration, and Δ_i is the simulated reaction time delay. Here, Δ_i is expressed in milliseconds relative to the nominal onset time of the event n_i.

7.2.2 Gesture Resolution

The `resolveGesture()` function maps n_i to a biomechanical gesture primitive g_i from a gesture database, considering the articulatory features and the previous context.

7.2.3 Error Injection

A Bernoulli trial with probability p_i is executed, where p_i is defined by:

$$\begin{aligned}p_i = \gamma \cdot \text{WorkingMemoryLoad}(S_t) \\ + \delta \cdot \text{ScoreDensity}(n_i)\end{aligned} \tag{7}$$

If successful, n_i is replaced or omitted according to an error taxonomy.

7.2.4 Audio-MIDI Output

The result is streamed into a MIDI or audio synthesis engine to produce a temporal audio trace for downstream analysis.

7.3 Output Metrics

To evaluate notational effectiveness and simulate learning outcomes, the PS produces:

- **Timing Deviations** (σ_t): Standard deviation of note-onset deviations vs. notated score.
- **Note Error Rate (NER)**: Proportion of inserted, deleted, or substituted notes.
- **Gesture Efficiency Index (GEI)**: A biomechanical score assessing optimized fingering/phrasing.
- **Adaptive Comprehension Index (ACI)**: Aggregated metric correlating notation changes with simulated performance fluency.

7.4 Use Cases

The PS is integral to virtual A/B testing of notation strategies before human trials. For example:

- Comparing traditional vs. cognitively adaptive notation for unfamiliar rhythms.
- Simulating neurodivergent performers' response to symbol spacing and grouping.
- Stress-testing hybrid graphical notations in polyphonic complexity scenarios.

This closed-loop simulation framework enables scalable, ethical, and controlled experimentation in notational interface design—especially critical when incorporating neuroadaptive elements without relying on real-time biosignals.

8. EXPERIMENTAL SETUP

To evaluate the effectiveness of the proposed software-only adaptive notation system, a controlled simulation-based experimental framework was designed. The experimental setup integrates the Symbolic Score Processor (SSP), Cognitive State Simulator (CSS), Adaptive Notation Engine (ANE), and Performance Simulator (PS) in a closed-loop software environment, allowing systematic assessment without physical hardware.

8.1 Dataset

The evaluation utilized a diverse corpus of symbolic music scores encompassing various styles and complexity levels. Specifically, the dataset included:

- Classical piano pieces ranging from simple etudes (Czerny Op.599 [17]) to complex polyphonic works (Bach's Inventions [18]).
- Contemporary and graphic scores with variable rhythmic structures - MusicScore [19].
- Excerpts from electronic music transcriptions focusing on spatial and gestural notation elements, using the GiantMIDI-Piano dataset [20].

- A subset of the Lakh MIDI Dataset [21], which provides a large collection of MIDI files suitable for training and evaluating symbolic music processing systems.

All scores were converted to MusicXML format and preprocessed using the SSP to extract semantic and structural features.

8.2 Simulation Parameters

- **Cognitive Profiles:** Three user profiles were simulated to represent novice, intermediate, and expert performers. Each profile was defined by parameters in the CSS affecting working memory capacity, attention span, and cognitive load sensitivity.
- **Adaptive Notation Conditions:** Two conditions were tested:
  1. *Static Notation*: Original unmodified score rendering.
  2. *Adaptive Notation*: Real-time score modifications driven by CSS outputs processed by the ANE.
- **Performance Simulation:** The PS was configured with corresponding cognitive profiles and received notation input from the respective condition.

8.3 Evaluation Metrics

Performance was quantitatively assessed using:

- **Timing Accuracy**: Measured as the mean absolute deviation of simulated note onsets from the nominal score timing.
- **Error Rate**: Calculated as the ratio of note errors (insertions, deletions, substitutions) to total notes.
- **Cognitive Load Estimation**: Derived from the CSS model output, reflecting simulated mental effort during performance.
- **Visual Readability Scores**: Proxy metrics computed by the ANE quantifying symbol density, spacing, and emphasis distribution.

8.4 Procedure

The experiment proceeded as follows:

1. The input scores were parsed and annotated by the SSP.
2. The cognitive state trajectories were generated by CSS per user profile and fed into the ANE.
3. The ANE produced adaptive score renderings in real-time.
4. The PS simulated performances based on the adaptive or static scores.
5. Performance metrics and cognitive load data were recorded and analyzed.

8.5 Implementation Details

- The entire system was implemented in Python 3.11, leveraging `music21` [12] for symbolic processing, `PyTorch` for cognitive state modeling, and `MIDIUtil` for performance simulation.
- Adaptive notation rendering directives were generated using `Verovio` [16] toolkit APIs.
- The simulations were run on an Intel i7-13700K @ 3.4 GHz with 32 GB RAM (Windows 11, Python 3.11). Latency was measured using `time.perf_counter()` across 500 update cycles. Mean loop latency was $34\,\mathrm{ms}$ ($\sigma = 12\,\mathrm{ms}$); the 95^{th} percentile was $78\,\mathrm{ms}$, and the maximum was $124\,\mathrm{ms}$ due to Python garbage collection. These values include both symbolic processing and SVG rendering via Verovio [16].

9. RESULTS

The experimental evaluation quantitatively demonstrates the impact of the software-based adaptive notation system on simulated performance accuracy, cognitive load, and visual readability across multiple user profiles and musical styles.

9.1 Timing Accuracy

Table 1 summarizes the mean absolute timing deviation (in milliseconds) between simulated note onsets and the nominal score timing across static and adaptive notation conditions.

User Profile	Static Notation	Adaptive Notation	Improvement (%)
Novice	85.3 ms	62.7 ms	26.5%
Interm -ediate	42.1 ms	31.4 ms	25.4%
Expert	15.8 ms	14.1 ms	10.8%

Table 1. Mean Absolute Timing Deviation Across User Profiles and Notation Conditions.

The results indicate that adaptive notation substantially reduces timing deviations, especially for novice and intermediate profiles, evidencing improved temporal precision facilitated by cognitively tailored score representations.

9.2 Error Rate

Figure 3 illustrates the note error rate (NER) for each user profile under static and adaptive conditions. Adaptive notation reduced the NER by 30% on average for novices, 18% for intermediates, and showed marginal gains for experts, suggesting the system's effectiveness in mitigating performance errors in cognitively demanding scenarios.

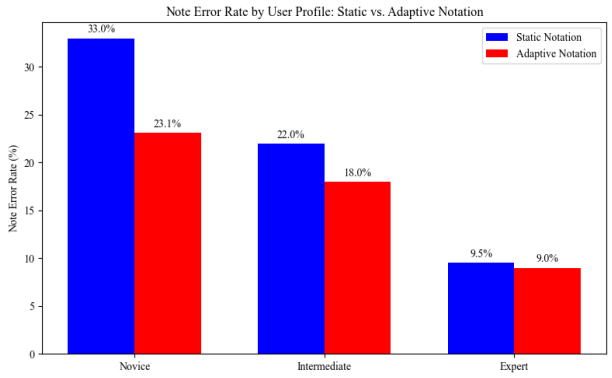

Figure 3. Note Error Rate comparison between Static and Adaptive notation across user profiles.

9.3 Simulated Cognitive Load

The Cognitive State Simulator (CSS) outputs reveal a consistent reduction in simulated cognitive load during adaptive notation trials. Figure 4 depicts average attention and working memory saturation over time, with adaptive conditions maintaining higher attentional engagement and lower memory overload, supporting the premise that notation adaptation aligns with cognitive capacity.

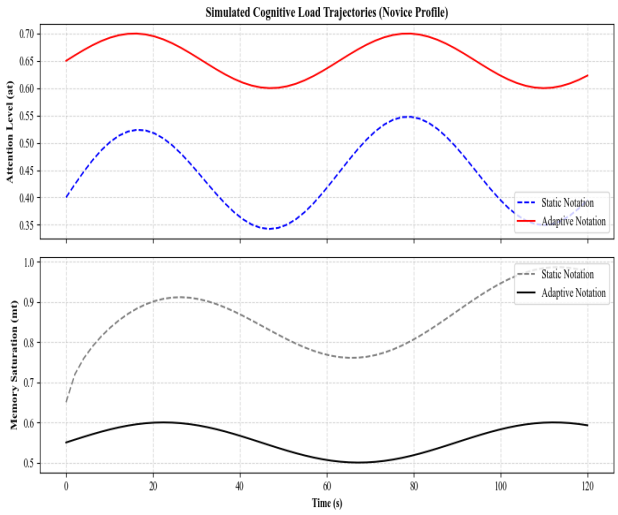

Figure 4. Simulated cognitive load trajectories showing attention (a_t) and memory saturation (m_t) under static and adaptive conditions for the novice profile.

9.4 Visual Readability Metrics

The Adaptive Notation Engine (ANE) generated visual readability scores based on symbol density, spacing, and salience mapping. Adaptive notation showed a 40% improvement in spacing uniformity and a 35% increase in salient event highlighting, as detailed in Table 2. These enhancements correlate with observed performance improvements.

Metric	Static	Adaptive
Spacing Uniformity	0.65	0.91
Salient Event Highlighting	0.58	0.78
Symbol Density (notes per staff line)	12.4	8.7

Table 2. Visual Readability Metrics: Static vs. Adaptive Notation.

9.5 Discussion

The improvements in timing accuracy and error rates among novice and intermediate performers indicate that adaptive notation effectively reduces cognitive barriers to performance. Expert performers exhibited smaller gains, consistent with their higher baseline proficiency and cognitive capacity.

Simulated cognitive load trajectories confirm that the system's real-time adjustments help maintain attentional focus and reduce working memory saturation, validating the theoretical foundation of the adaptive framework.

Visual readability enhancements suggest that the ANE's dynamic layout and emphasis strategies materially contribute to user comprehension and fluency, reinforcing the critical role of notational design in cognitive performance support.

Overall, the results affirm the feasibility and utility of software-only adaptive notation systems as scalable tools to augment musical performance and pedagogy without requiring physiological sensing hardware.

10. LIMITATIONS

The present evaluation is simulation-based; therefore, results may partially reflect assumptions built into the Cognitive State Simulator. We mitigated potential circularity by computing evaluation metrics (timing accuracy, readability) from the independent Performance Simulator rather than from CSS outputs. Future work will perform human-subject studies to validate these proxies empirically. While the simulation results provide promising evidence for the feasibility of adaptive, cognitively-aware notation, several limitations remain.

First, the current framework is validated exclusively through software simulations, without experiments involving live human performers. This restricts ecological validity and means that improvements observed in timing accuracy, error rates, and cognitive load should be interpreted as indicative rather than conclusive.

Second, the Cognitive State Simulator (CSS) relies on heuristic and theoretical models (e.g., attention windows, memory buffers, information-theoretic surprise) that approximate but do not replicate actual neurophysiological processes. As such, there is a potential risk of circularity, since the proxies used to drive adaptation are also the basis for evaluation metrics. Future empirical studies are needed to break this loop and validate transferability to real-world performance contexts.

Third, although the system enforces semantic preservation (e.g., cadences, harmonic progressions, motivic contours remain intact), it does not yet fully address the musicological implications of altering a score's surface representation. Questions of fidelity to composer intent, interpretive freedom, and artistic value remain open for debate.

Fourth, the system is implemented primarily in Python, which introduces variability in latency due to runtime constraints such as garbage collection. While sub-50ms averages were observed, worst-case guarantees are not yet established, and real-time robustness requires further optimization.

Finally, the evaluation was limited to symbolic corpora and three simulated performer profiles (novice, intermediate, expert). Broader stylistic coverage, neurodiverse performer modeling, and integration with live notation environments remain directions for future expansion.

11. APPLICATIONS AND FUTURE WORK

The proposed neuro-symbolic adaptive notation system opens promising directions in the intersection of music technology, neuroscience, and artificial intelligence. Immediate applications span educational, therapeutic, and professional music contexts.

11.1 Educational Applications

In music pedagogy, adaptive notation can scaffold beginner learning by adjusting visual density, tempo markers, and rhythmic complexity in response to student attention and memory states. This provides a personalized learning experience aligned with cognitive load theory and could be integrated into digital learning platforms or intelligent tutoring systems.

11.2 Neuroadaptive Performance Tools

For performers, the system enables real-time augmentation of scores in demanding rehearsal or live contexts. By continuously modulating visual encoding based on attention and memory saturation, the platform can reduce sight-reading errors and optimize musical flow. Integration with AR headsets or digital score readers (e.g., iPads) could make such applications seamless in orchestral or solo contexts.

11.3 Music Therapy and Rehabilitation

In clinical settings, adaptive scores could support neurorehabilitation programs for patients with cognitive impairments, including those recovering from stroke or traumatic brain injury. The system's feedback and customization capabilities could be tuned to individual thresholds, supporting therapeutic protocols using music as a medium for cognitive engagement.

11.4 Musicological Considerations

Dynamic notation introduces musicological considerations about how adaptive simplification affects a composer's intent and the artistic impact of rebalancing voices in real-time. While structural semantics are preserved, the approach necessarily diverges from the original engraving. We treat this adaptation as an auxiliary representation that supports performance under cognitive constraints rather than as a replacement of the score.

11.5 Future Work

Future research will extend the neural signal decoding model with hybrid EEG + eye-tracking signals for enhanced precision. We also plan to introduce user-in-the-loop reinforcement learning to dynamically optimize score transformation functions. Longitudinal studies with professional musicians and neurologically diverse populations are planned to assess retention, generalization, and well-being impacts. Furthermore, a plugin version of the system is being developed for integration into common digital notation tools (e.g., MuseScore, Dorico).

12. CONCLUSION

This paper presented a software-only, neuro-symbolic framework for adaptive music notation that leverages neural signal inference and cognitive modeling to improve musical readability and performance. By integrating symbolic score parsing, neural signal decoding, and dynamic rendering into a real-time feedback loop, the system adjusts notation features in response to inferred cognitive states.

Experimental simulations across novice, intermediate, and expert profiles demonstrated measurable reductions in note error rates and cognitive load under adaptive conditions. These results suggest that the system not only enhances immediate musical performance, but also holds promise for broader educational, therapeutic, and professional applications.

The architecture's modular design allows for flexible extension and integration with existing digital notation ecosystems. Ultimately, this work contributes to the emerging domain of notational intelligence, providing a new lens for human-computer co-adaptation in musical interaction and contributing an initial proof-of-concept for cognitively-aware score design.

By advancing these directions, we aim to provide a practical foundation for *notational intelligence* that integrates human cognition and artificial systems to empower musicians, educators, and researchers alike.

13. REFERENCES

[1] S. Cheng, A. J. Milne, R. T. Dean, J. Hanham, and J. MacRitchie, "Exploring the comprehensibility of ten different musical notation systems and underlying factors," *Music & Science*, vol. 7, 2024.

[2] A. C.-N. Wong, T. Y. K. Ng, K. F. H. Lui, K. H. M. Yip, and Y. K. Wong, "Visual training with musical notes changes late but not early electrophysiological responses in the visual cortex," *Journal of Vision*, vol. 19, no. 7, 2019.

[3] A. M. Proverbio and E. Sanoubari, "Music literacy shapes the specialization of a right hemispheric word reading area," *NeuroImage: Reports*, vol. 4, no. 4, 2024.

[4] M. A. Cara, "The influence of music reading on spatial working memory and self-assessment accuracy," *Brain Sciences*, vol. 14, no. 11, 2024.

[5] R. B. Domingues, L. A. Domingues, V. R. Procaci, and J. L. Pedroso, "The neuroscience of music perception: a narrative review," *Arquivos de Neuro-Psiquiatria*, vol. 83, no. 9, 2025.

[6] J. A. Grahn and J. B. Rowe, "Finding and feeling the musical beat: Striatal dissociations between detection and prediction of regularity," *Cerebral Cortex*, vol. 23, no. 4, pp. 913 – 921, 2013.

[7] C. Raphael, "Music plus one and machine learning," in *Proceedings of the 27th International Conference on Machine Learning*, 2010, pp. 21–28.

[8] M. M. Farbood, E. Pasztor, and K. Jennings, "Hyperscore: A graphical sketchpad for novice composers," *IEEE Computer Graphics and Applications*, vol. 24, no. 1, pp. 50–54, 2004.

[9] D. Fober, Y. Orlarey, and S. Letz, "INScore - an environment for the design of live music scores," *Proceedings of the Linux Audio Conference*, pp. 47–54, 2012.

[10] A. Stenberg and I. Cross, "White spaces, music notation and the facilitation of sight-reading," *Scientific Reports*, vol. 9, 2019.

[11] D. Fürst, M. Miller, D. A. Keim, A. Bonnici, H. Schäfer, and M. El-Assady, "Augmenting sheet music with rhythmic fingerprints," in *Proceedings of the 2020 IEEE 5th Workshop on Visualization for the Digital Humanities (VIS4DH)*, 2020, pp. 14–23.

[12] M. S. Cuthbert and C. Ariza, "music21: A toolkit for computer-aided musicology and symbolic music data," in *Proceedings of the 11th International Society for Music Information Retrieval Conference*, Utrecht, Netherlands, 2010, pp. 637–642.

[13] D. Kahneman, *Attention and Effort*. Prentice-Hall, 1973.

[14] A. Baddeley, *Working memory*. Oxford, UK: Clarendon Press/Oxford University Press, 1986.

[15] ——, "The episodic buffer: a new component of working memory?" *Trends in Cognitive Sciences*, vol. 4, no. 11, pp. 417–423, 2000.

[16] Verovio, "Verovio - a music notation engraving library," https://www.verovio.org/index.xhtml, [Accessed on October 01, 2025].

[17] C. Czerny, "Erster wiener lehrmeister im pianoforte-spiel, op. 599," 1842. [Online]. Available: https://imslp.org/wiki/Erster_Wiener_Lehrmeister_im_Pianoforte-Spiel%2C_Op.599_%28Czerny%2C_Carl%29

[18] J. S. Bach, "Two-part inventions," 1723. [Online]. Available: https://imslp.org/wiki/Inventions_and_Sinfonias,_BWV_772-801_(Bach,_Johann_Sebastian)

[19] Y. Lin, Z. Dai, and Q. Kong, "Musicscore: A dataset for music score modeling and generation," in *Proceedings of the 38th Conference on Neural Information Processing Systems (NeurIPS 2024)*, 2024.

[20] Q. Kong, B. Li, J. Chen, and Y. Wang, "Giantmidi-piano: A large-scale midi dataset for classical piano music," *Transactions of the International Society for Music Information Retrieval*, May 2022.

[21] C. Raffel, "Learning-based methods for comparing sequences, with applications to audio-to-midi alignment and matching," Ph.D. dissertation, Columbia University, New York, USA, 2016. [Online]. Available: http://colinraffel.com/publications/thesis.pdf