

Enabling Semantic-Aware Networked Music Performance with AI Inference at the Edge

Luca Borgianni¹, Gaetano Francesco Pittalà², Gianluca Davoli², Stefano Giordano¹

¹University of Pisa, Dept. of Information Engineering, Pisa, Italy

²University of Bologna, Dept. of Electrical, Electronic, and Information Engineering, Bologna, Italy

Corresponding author: luca.borgianni@unipi.it

Abstract—Networked Music Performance (NMP) systems enable geographically distributed musicians to collaborate in real time, yet they remain fundamentally limited by semantic blindness, i.e., the inability to interpret the musical content of the audio stream being transmitted. This paper moves toward *Semantic NMP* by showing that audio inference deployed at the Edge can be integrated in NMP pipelines without violating real-time constraints. To realize this vision, we design and implement a modular framework integrating three complementary analysis engines that operate concurrently on the live stream: instrument recognition via Pretrained Audio Neural Networks, librosa-based rhythmic feature extraction, and spectral quality monitoring, all embedded within a JackTrip-based NMP workflow. We evaluate the full pipeline under three deployment strategies (local, edge-assisted, and cloud-based) and quantify their impact on inference latency. Under local execution, processing averages 28 ms per chunk (38.5 ms total across all three engines). Cloud deployment yields 56.7 ms inference, while edge deployment increases neural inference to 106.3 ms but confines transport latency to 0.1–1 ms. As semantic analysis runs asynchronously from the audio path, these delays affect semantic-update latency rather than the NMP transport budget. Edge computing thus offers a favorable latency-computation trade-off for semantic-aware NMP.

Index Terms—Networked Music Performance, Edge Computing, Semantic Audio Analysis, Instrument Recognition, Distributed Music Systems

I. INTRODUCTION

Networked Music Performance (NMP) systems enable geographically distributed musicians to perform together in real time, pushing the limits of ultra-low latency networking and interactive media processing. Achieving a seamless musical interaction requires stringent end-to-end delay constraints, typically below 20–30 ms, beyond which human perception degrades synchronization and musical expressiveness [1]. As a result, NMP has historically focused on optimizing transport protocols, audio encoding pipelines, and network paths to minimize latency and jitter. Yet, despite these advances, a fundamental limitation persists: NMP systems remain *semantically blind*, transporting audio faithfully but without any understanding of its musical content.

At the same time, recent advances in Artificial Intelligence (AI), particularly in audio understanding and real-time inference, have opened new opportunities for augmenting musical interaction. Tasks such as instrument recognition, rhythmic analysis, and audio quality monitoring can provide *semantic awareness* of the music stream, mapping raw audio signals to high-level descriptors that capture *what* is being played, *how* it

is structured, and *whether* it is degrading. This semantic layer enables adaptive effects, intelligent mixing, and context-aware enhancements, representing a qualitative shift from mere audio transport to musically-informed network behavior. However, integrating AI into NMP systems introduces a fundamental challenge: *where should the intelligence be placed in the network to preserve real-time constraints?*

Traditional cloud-centric approaches offer scalable computational resources, but introduce additional delays that may violate the strict latency requirements of NMP. On the other hand, fully local processing at the client side is constrained by limited computational capabilities and energy budgets, especially in lightweight or embedded setups. Edge computing emerges as a promising middle ground, offering low-latency processing close to the user while retaining sufficient computational power for AI inference. Crucially, the edge is not merely a latency optimization: it is the enabling layer that makes semantic awareness compatible with real-time musical interaction, allowing semantic descriptors to be extracted and acted upon before the audio stream is forwarded further. Despite its potential, the feasibility of deploying AI-driven semantic audio processing within NMP systems remains largely unexplored.

This paper addresses this gap by investigating the integration of AI-based audio analysis within NMP architectures, with a specific focus on edge-assisted deployment. Rather than proposing a new networking paradigm, we aim to answer a more fundamental and practical question: can *semantic awareness* be embedded into real-time musical interaction without interfering with the low-latency audio transport path, and if so, where should semantic processing be placed to provide timely and predictable descriptors?

To this end, we design and implement a modular framework that incorporates multiple real-time audio analysis models, including instrument recognition, rhythmic feature extraction, and signal quality monitoring, operating on live audio streams. We evaluate different deployment strategies (local, edge-based, and cloud-based), quantifying their impact on end-to-end latency and real-time feasibility within a representative NMP workflow based on JackTrip.

The contributions of this work are threefold:

- **Multi-Modal Semantic Framework:** We design and implement a modular framework that integrates heterogeneous AI-driven audio analysis tasks.

- **Architectural Deployment Exploration:** We define and formalize multiple deployment strategies, including local, cloud-centric, and edge-assisted execution, and introduce JackTrip-as-a-Service (JT-aaS) as a conceptual edge-native extension for future managed deployments.
- **Experimental Validation:** We characterize the computational cost of the semantic pipeline under local, cloud, and edge execution and show that all analysis modules operate comfortably within the adopted 2-second semantic processing window. The results further highlight the trade-off between faster cloud computation and lower, more predictable network overhead at the edge.

The remainder of this paper is organized as follows. Section II reviews the related work on NMP systems and AI-driven audio analysis. Section III describes the proposed semantic-aware NMP architecture under cloud and edge deployment models. Section IV details the multi-modal audio processing pipeline, the semantic mapping layer, and the real-time implementation constraints. Section V defines the four experimental deployment scenarios considered in the evaluation. Section VI presents and discusses the experimental results, covering computational performance, feature consistency, semantic discriminative power, and long-term stability. Section VII concludes the paper and outlines directions for future work.

II. RELATED WORK

A. NMP Architectures and Latency Constraints

Networked Music Performance (NMP) enables geographically distributed musicians to collaborate in real time over data networks [1]. Early pioneering work by the SoundWIRE group at Stanford University established the technical foundations of interactive audio streaming over the Internet, emphasizing the fundamental barriers imposed by network delay [2], [3]. A key limiting factor is the Ensemble Performance Threshold (EPT) [4], which defines the strict upper boundary for synchronous acoustic interaction. Empirical studies indicate that human perception degrades performance synchronization when the end-to-end delay exceeds 20–30 ms [1], with values above 25 ms severely disrupting rhythmic coordination and ensemble cohesion.

To satisfy the EPT, substantial research has targeted low-latency systems and transport protocols. Open-source frameworks like JackTrip and LOLA minimize transmission delay by utilizing uncompressed audio streaming over bare UDP combined with ultra-low buffer sizes [5]. While effective under stable corporate or academic networks, these architectures remain highly vulnerable to volatile unmanaged public Internet conditions.

B. Network-Aware Optimization and Advanced Infrastructures

Beyond transport protocol tuning, extensive literature addresses the impact of heterogeneous network impairments on distributed performances. Packet jitter is often as disruptive as absolute delay, causing timing instability and degrading the user’s Quality of Experience (QoE) [6]. Traditional mitigation

strategies rely on network-aware audio encoding [7], forward error correction, and adaptive playout buffering [8].

More recently, next-generation network infrastructures have expanded the geographic reach of NMPs. Frameworks leveraging 5G cellular systems have enabled localized untethered musical interaction [9], while Low Earth Orbit (LEO) satellite constellations like Starlink have connected remote locations [10]. Nevertheless, despite high bandwidth capacity, LEO satellite links introduce variable latency profiles that continuously challenge the rigid deterministic boundaries imposed by the EPT.

C. AI Integration and Edge Processing in Multimedia Streams

In parallel, Artificial Intelligence (AI) has been introduced into the NMP domain, primarily focused on signal-level enhancements. Deep learning models are increasingly utilized for packet loss concealment, predictive network modeling, and real-time audio frame reconstruction under adverse conditions [11], [12]. Concurrently, AI-driven musical interaction explores real-time accompaniment and human–AI co-performance systems [13].

Although these approaches demonstrate the potential of intelligent audio processing, they rarely consider where inference should be executed and how deployment architectures impact real-time transport performance. Centralized cloud-based inference introduces additional internet-backbone delays incompatible with strict interactive constraints. To solve this, Edge Computing has emerged as a promising paradigm to offload media processing closer to the end users, dramatically compressing inference delay. However, while edge-assisted processing is widely investigated in generic real-time multimedia workflows, its systemic evaluation within high-fidelity, ultra-low-latency NMP environments remains severely limited, particularly as emerging spatial and immersive audio environments introduce steep computational footprints [14].

D. Research Gap and Contribution

A clear gap emerges from the surveyed literature: existing NMP architectures address isolated networking or signal-level optimization challenges without a comprehensive, multi-model evaluation under varying deployment layers. Furthermore, current systems remain semantically blind, transporting raw audio streams without any high-level contextual understanding of the musical interaction.

To bridge this gap, this paper introduces a *semantic-aware NMP framework*. Rather than shifting away from raw audio streaming, our architecture augments traditional JackTrip pipelines by extracting concurrent, high-level semantic metadescription. Unlike prior works that focus on single-task AI enhancements, we evaluate heterogeneous semantic workloads under strict EPT constraints and compare deployment layers systematically. We evaluated multiple AI-based analysis models under heterogeneous NMP deployment scenarios (local, cloud, and edge-assisted), providing a formal system-level perspective on the computational trade-offs of intelligent distributed music systems.

III. PROPOSED SEMANTIC-AWARE NMP ARCHITECTURE

This section describes the two primary deployment models evaluated within the semantic-aware NMP framework. For consistency with the implementation and the diagrams, we retain the label *IR_service*; however, the service encapsulates the complete multi-modal semantic pipeline, including instrument recognition, rhythmic analysis, and quality monitoring.

A. Cloud-Centric Deployment Model

The first architecture hosts the audio processing and semantic analysis service on a centralised cloud infrastructure, as illustrated in Fig. 1. When a musician initiates a performance session via the local JackTrip client, a service request is forwarded to an Edge Orchestrator, which authenticates the session and provisions a containerised Instrument Recognition service (*IR_service*) on the remote cloud endpoint.

Once the cloud instance is active, the data plane follows a multi-hop path:

- **Stream forwarding:** The audio stream handled by the JackTrip endpoint is mirrored and transmitted to the cloud-hosted semantic analysis service.
- **Semantic inference:** The cloud-resident *IR_service* processes the incoming audio chunks, executing the parallel neural inference and feature extraction pipelines.
- **Contextual filtering:** Adaptive audio filters are applied based on the detected instrument taxonomy and signal quality profile.
- **Stream return:** The processed audio stream is routed back to the client application across the network backbone.

B. Edge-Assisted Deployment Model

The second architecture relocates the semantic processing service to a node at the network edge, as depicted in Fig. 2. The control path is identical to the cloud model: the Edge Orchestrator receives the session request and provisions the *IR_service* container. However, instead of deploying to a remote data centre, the orchestrator instantiates the service on a local edge server situated in close proximity to the client's access network.

Co-locating computation with the access node confines the data plane to the edge domain:

- **Local routing:** The incoming audio stream is routed directly to the edge computing node, bypassing the internet backbone.
- **Edge inference:** The *IR_service* performs instrument recognition, feature aggregation, and semantic classification within the edge domain.
- **Low-latency return:** The filtered audio track is returned to the JackTrip client without traversing the core network.

C. JackTrip-as-a-Service Architecture

A third deployment variant, introduced in this work as *JackTrip-as-a-Service* (JT-aaS), extends the edge model by migrating the JackTrip server itself to the edge node alongside the *IR_service*. Under this configuration, the orchestration

layer handles per-user session authentication and instantiation, eliminating the need for a dedicated JackTrip client process on the end device. More importantly, inference and filtering are applied directly to the audio stream at the server side, before it is forwarded to the user.

IV. EXPERIMENTAL METHODOLOGY

The proposed multi-modal analysis framework was implemented and validated using a modular Python-based architecture. The system consists of three specialised engines, each designed for a distinct analytical task and operating concurrently on a shared stream of discrete audio chunks.

A. Audio Processing Pipeline

To ensure computational efficiency while satisfying the heterogeneous input requirements of the three models, the system operates on discrete, non-overlapping audio chunks extracted from the live stream. Processing parameters are configured independently per module:

- **Model A (Instrument Recognition):** Based on the PANNs (Pretrained Audio Neural Networks) architecture [15], which was pre-trained on the AudioSet corpus [16] comprising 527 sound classes. The input signal is resampled to $f_s = 32\text{ kHz}$ prior to feature extraction. Inference is accelerated via a CUDA-enabled device to minimise per-chunk processing time.
- **Model B (Rhythmic Analysis):** Employs *librosa*-based onset detection and beat tracking [17] operating at $f_s = 48\text{ kHz}$. Onset events and RMS energy are computed per chunk to derive rhythmic density estimates and section-level labels.
- **Model C (Quality Monitoring):** Also operates at $f_s = 48\text{ kHz}$. Spectral flatness and spectral centroid are extracted per chunk as proxy indicators of signal degradation, codec artefacts, and clipping.

It is worth emphasising that the primary objective of this work is not to provide an exhaustive comparative evaluation of the three AI models in isolation, nor to benchmark their classification accuracy against state-of-the-art alternatives. We intentionally select three representative models as representative instances of semantically distinct analysis tasks chosen to demonstrate how heterogeneous AI workloads can be embedded at different points of the NMP architecture and evaluated under different deployment strategies.

B. Semantic Mapping and Classification

A key contribution of the framework is the translation of low-level acoustic features into high-level semantic descriptors that are interpretable at the application layer. Two complementary mapping layers are defined, one for instrument identity and one for rhythmic texture.

1) *Instrument Taxonomy:* The raw output of Model A is a 527-dimensional probability vector over all AudioSet classes. Rather than using this fine-grained label space directly, we implement a custom aggregation layer that groups semantically related classes into ten functional macro-categories suited to

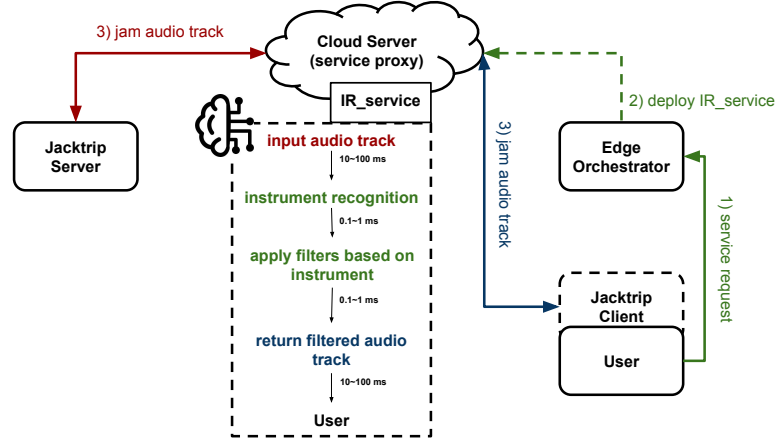


Fig. 1. Workflow for the evaluated Cloud provisioning of services.

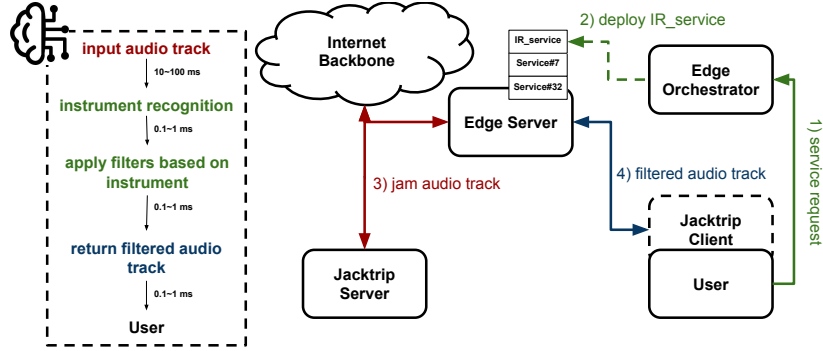


Fig. 2. Workflow for the evaluated Edge provisioning of services.

the NMP context. Representative groupings are reported in Table I. For each macro-category k , a binary activation flag $a_k \in \{0, 1\}$ is derived as:

$$a_k = \mathbf{1} \left[\max_{j \in \mathcal{C}_k} p_j \geq \tau \right] \quad (1)$$

where $\mathcal{C}_k$ is the set of AudioSet label indices belonging to category k , p_j is the model's output probability for label j , and $\tau = 0.2$ is a confidence threshold. The value $\tau = 0.2$ was selected empirically to balance recall (avoiding missed detections on instruments with naturally diffuse spectral profiles, such as strings) against precision (suppressing spurious activations in silence or noise segments).

2) *Rhythmic Semantics*: Model B derives two complementary rhythmic descriptors per chunk. First, the *beat rate* is estimated as the number of detected onset events per second. Second, the RMS energy is computed as a measure of dynamic intensity. The joint distribution of these two quantities is used to assign each chunk to one of four section-level categories: *dense groove* (high beat rate, high energy), *sparse groove* (low beat rate, moderate energy), *onset bursts* (high beat rate, low sustained energy), and *other*.

TABLE I
INSTRUMENT MACRO-CATEGORIES FOR MODEL A OUTPUT AGGREGATION.

Macro-category	Representative AudioSet labels
Drums	Drum kit, Snare drum, Bass drum, Cymbal
Bass	Bass guitar, Double bass
Strings	Violin, Cello, String section
Vocals	Singing, Choir, Vocal
Piano	Piano, Electric piano, Keyboard
Electric Lead	Electric guitar, Distortion guitar
Acoustic Guitar	Acoustic guitar, Fingerpicking
Brass & Wind	Trumpet, Saxophone, Flute
Percussion	Bongo, Claves, Maracas
Other	Synthesizer, Organ, Harp

C. Real-time Implementation Constraints

Live audio acquisition was performed using the `sounddevice` Python API, which provides low-latency access to the system audio interface and enables direct integration with the JackTrip audio stream. To guarantee continuous operation without buffer overflow, each processing iteration must satisfy the following timing constraint:

$$T_{\text{inference}} + T_{\text{overhead}} < T_{\text{chunk}} \quad (2)$$

where $T_{\text{inference}}$ is the total inference time across all three models for a single chunk, T_{overhead} accounts for data transfer,

resampling, and feature formatting operations, and T_{chunk} denotes the chunk acquisition window. In all experiments T_{chunk} was set to 2.0s, chosen as a practical trade-off between the temporal resolution of the semantic descriptors and headroom for inference.

It is important to clarify the relationship between the 2-second analysis window and the real-time latency constraints of the NMP transport layer. The audio stream transmitted between JackTrip endpoints operates at the standard NMP latency of 20–30ms and is never delayed or buffered for semantic analysis purposes. The multi-modal analysis pipeline runs *asynchronously* and in parallel with the transport layer: it continuously processes non-overlapping 2-second windows of the audio stream to produce semantic descriptors, which are then applied to *subsequent* audio packets. This introduces a *semantic update latency* of at most $T_{\text{chunk}} = 2.0\text{s}$ but imposes no additional delay on the audio transport itself. In practice, this latency is perceptually acceptable for the application-layer services targeted by this work (instrument-aware filtering, adaptive mixing, quality-triggered rerouting), as these operate on the timescale of musical phrases rather than individual audio packets.

It should be noted that the present study focuses on the feasibility and latency implications of semantic descriptor extraction within the NMP pipeline, rather than on the design or evaluation of specific audio filters. The *contextual filtering* step represents a target application layer that consumes the semantic descriptors produced by the IR service; its computational cost and perceptual effects are left for future work.

V. EXPERIMENTAL EVALUATION

This section presents the experimental evaluation of the proposed framework across four deployment scenarios, designed to quantify the latency overhead introduced by the IR service under different computational placement strategies.

A. Evaluation Scenarios

Four deployment scenarios are considered at the architectural level. The experimental measurements reported in this work focus on the computational behavior of the local, cloud, and edge semantic-processing configurations, while JT-aaS is included as a conceptual deployment extension.

Scenario 1 – Baseline JackTrip. JackTrip runs on the client device as intended, with no additional processing service. End-to-end latency measurements are collected on the client side and serve as the baseline against which the overhead of the AI service is assessed.

Scenario 2 – Edge-assisted Instrument Recognition. JackTrip continues to run on the client, but the received audio stream is additionally forwarded to an IR service deployed on an edge node within the client’s local network. The processing pipeline is as follows: (i) the audio stream is received by the JackTrip client; (ii) the stream is sent to the local edge node; (iii) instrument recognition is performed and a filter is applied; (iv) the processed stream is returned to the client.

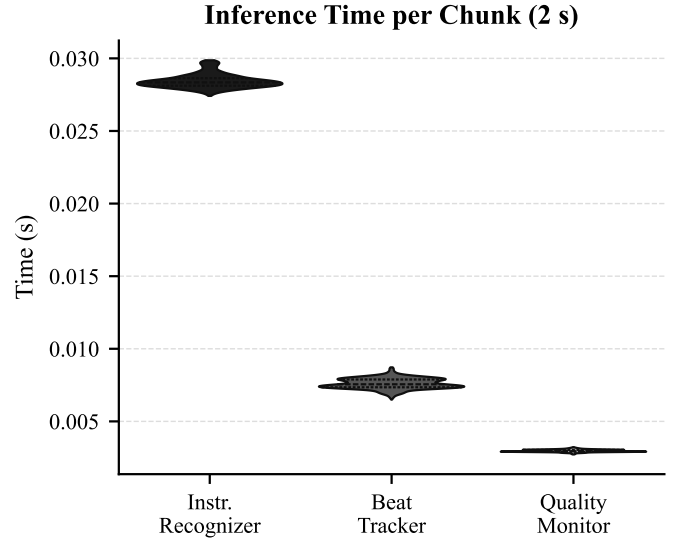


Fig. 3. Distribution of computational inference times per 2-second audio chunk across the three specialised modules. Model A (Instrument Recognizer) averages approximately 28 ms, while Model B (Beat Tracker) and Model C (Quality Monitor) exhibit negligible latencies well under 10 ms, demonstrating real-time viability.

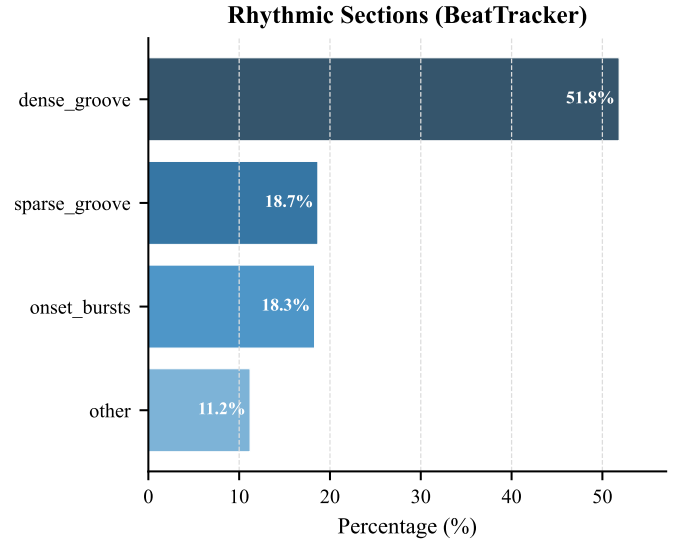


Fig. 4. Distribution of rhythmic sections identified by the BeatTracker module over the full 2,000-chunk evaluation excerpt. The spread across all four categories confirms that the taxonomy captures genuine variation in rhythmic texture without collapsing to a dominant class.

Scenario 3 – Cloud-assisted Instrument Recognition. This scenario mirrors Scenario 2, but the IR service is hosted on a remote cloud infrastructure rather than a local edge node. The audio stream is therefore routed through the Internet backbone to reach the processing endpoint.

Scenario 4 – JackTrip-as-a-Service (JT-aaS). In this scenario, the JackTrip server itself is migrated to the edge node and exposed as a managed service. An orchestration layer handles authentication and per-user session instantiation,

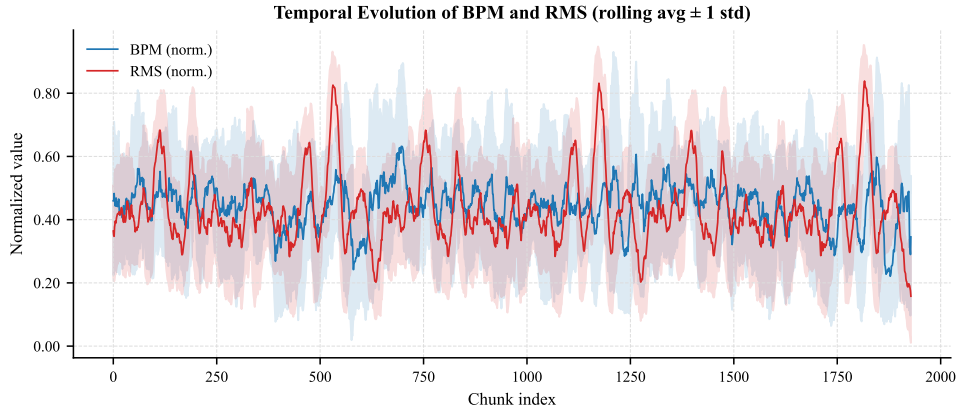


Fig. 5. Temporal evolution of normalized BPM and RMS energy over 2,000 consecutive audio chunks, smoothed via a rolling average with ± 1 standard deviation shading. The partial decorrelation between the two descriptors confirms their complementary nature within the semantic representation.

so that no JackTrip client process is required on the end device. Inference and audio filtering are applied directly to the stream produced by the edge-resident JackTrip server before delivery to the user, eliminating the extra round-trip incurred in Scenarios 2 and 3.

The present experiments constitute a single-client feasibility study. Their purpose is to isolate the computational and placement costs of semantic analysis rather than to reproduce a complete multi-musician NMP session. Multi-client interaction, contention, and synchronization effects are therefore outside the scope of the present evaluation.

The framework was evaluated on a subset of the IRMAS dataset [18], a publicly available collection of music recordings annotated with predominant instrument labels. The selected excerpt totals 3,859 s (48 kHz, mono) and spans multiple instrument categories and musical genres, providing sufficient variety in instrumentation and rhythmic texture to exercise all three analysis modules under realistic conditions.

VI. RESULTS AND DISCUSSION

This section presents the experimental evaluation of the proposed semantic analysis framework. First, the standalone performance of the individual AI models is analyzed in terms of computational efficiency, semantic discriminative capability, and long-term operational stability. Subsequently, the results obtained under the different presented deployment scenarios are discussed, with particular emphasis on the impact of cloud, edge, and integrated JTaaS configurations on latency and scalability.

A. Computational Performance and Real-time Viability

The primary requirement for the semantic analysis framework is to maintain continuous processing within the acquisition window ($T_{\text{chunk}} = 2.0$ s). Fig. 3 illustrates the distribution of execution times for each specialized engine. The Instrument Recognizer (Model A), despite its deep neural architecture, achieves an average inference time of approximately 28 ms, operating well below the chunk duration. In contrast, the Beat Tracker (Model B) and the Quality Monitor (Model C)

exhibit even lower latencies, averaging roughly 7.5 ms and 3 ms, respectively.

These results confirm that the cumulative processing time per chunk (≈ 38.5 ms) is orders of magnitude lower than the 2.0 s acquisition window, as required by Eq. (2).

B. Long-term Stability and Quality Assessment

The temporal evolution of normalised BPM and RMS energy over the full 2,000-chunk sequence is shown in Fig. 5. Both descriptors exhibit continuous variation throughout the excerpt, reflecting genuine musical dynamics (tempo fluctuations, section transitions, and changes in performance intensity) rather than processing artefacts. Importantly, neither signal shows drift or systematic deviation over time, confirming that the pipeline does not accumulate estimation errors across chunks. The rolling average and standard deviation bands remain stable across the full duration, demonstrating that the system can sustain consistent semantic extraction over extended sessions without degradation. It is also worth noting the partial decorrelation between BPM and RMS energy visible in the plot: sections of high rhythmic activity do not always correspond to high energy, and vice versa.

The rhythmic section distribution reported in Fig. 4 shows that the dominant pattern in the evaluated material is *dense groove* (51.8%), followed by *sparse groove* (18.7%) and *onset bursts* (18.3%), with a residual *other* category accounting for 11.2%. The ability of the BeatTracker to distribute chunks across all four categories confirms that the taxonomy captures genuine variation in rhythmic texture and is not biased toward a degenerate solution. The near-equal occurrence of sparse groove and onset bursts indicates that the heuristic taxonomy does not trivially collapse these two categories on the evaluated excerpt. This analysis should be interpreted as a characterization of descriptor diversity rather than as an accuracy assessment, since no ground-truth rhythmic-section annotations are available.

C. Cloud vs. Edge Inference Time Comparison

Fig. 6 reports the per-chunk inference time of the three models across 150 consecutive chunks under both cloud and edge deployment.

Model A (Instrument Recognizer). Model A exhibits a significant performance gap between deployment environments: average inference time under cloud deployment is 56.7 ± 5.3 ms, while under edge deployment it rises to 106.3 ± 3.5 ms. The edge node therefore incurs a roughly $1.87\times$ higher inference cost for the neural component alone. The result is consistent with the hardware asymmetry between a managed cloud instance typically backed by server-grade CPUs or shared GPU accelerators, and a local edge node, which operates on more constrained hardware. However, the higher absolute latency on the edge does not compromise real-time viability: even at 106.3 ms, Model A operates well within the 2-second chunk acquisition window ($T_{\text{chunk}} = 2.0$ s), satisfying Eq. (2) with a headroom factor exceeding $18\times$. Furthermore, the edge trace exhibits a markedly lower variance ($\sigma = 3.5$ ms vs. 5.3 ms for cloud), indicating more predictable scheduling behaviour, a property that is arguably more important than raw throughput in a continuous streaming context.

Model B (Beat Tracker). The rhythmic analysis module follows a similar trend: edge deployment yields a higher average inference time (53.4 ± 1.9 ms) compared to the cloud infrastructure (22.0 ± 0.2 ms). While the lightweight, *librosa*-based onset detection and beat tracking pipeline is primarily CPU-bound, the constrained edge processor requires approximately $2.43\times$ more time to complete the analysis compared to the robust, server-grade CPU resources available in the cloud. Despite this absolute difference in processing speed, both environments maintain highly stable execution profiles, with the cloud trace exhibiting negligible variance ($\sigma = 0.2$ ms) and the edge trace confirming a predictable, deterministic execution suitable for real-time applications.

Model C (Quality Monitor). Model C exhibits the lowest absolute latency of all three engines under both strategies: 8.0 ± 0.2 ms on cloud and 5.0 ± 0.1 ms on edge. The negligible variance in both cases (≤ 0.2 ms) reflects the simplicity of spectral flatness and centroid computation, which imposes minimal resource demand regardless of the execution environment. The zero-variance edge result ($\sigma = 0.0$ ms at the reported precision) further indicates that this module is effectively I/O-bound rather than compute-bound, and would remain real-time viable even on highly constrained embedded hardware.

Cumulative latency and deployment trade-offs. Summing the per-model averages, the semantic processing cost is approximately 164.7 ms per chunk under edge deployment and 86.7 ms under cloud deployment. Thus, the cloud platform executes the multi-modal analysis more rapidly in raw computational terms. However, because the semantic pipeline runs asynchronously with respect to the JackTrip audio transport path, these inference times are not added to the audio-path latency and should not be compared directly with the 20–30

ms EPT. Deployment placement instead affects the timeliness and predictability of semantic updates: a cloud-hosted service additionally requires traversal of the Internet backbone, whereas edge placement confines this communication to the local access domain. Edge execution therefore trades higher computation time for lower and more predictable communication delay in the semantic-control loop.

Stability and transient behaviour. Both deployment environments exhibit a brief warm-up transient in the first 5–10 chunks, after which inference times stabilise to their steady-state distributions. This transient is consistent with JIT compilation and model cache warm-up effects typical of deep learning inference frameworks. The cloud trace for Model A shows isolated latency spikes of up to 77 ms at chunk 0 and approximately 68 ms at chunk 100, likely attributable to cloud resource preemption or network-induced scheduling delays. No comparable spikes are observed in the edge traces, reinforcing the stability advantage of localized computation for latency-sensitive streaming workloads.

In summary, the comparison suggests that edge deployment is an attractive strategy for semantic-aware NMP when timely and predictable semantic feedback is prioritized. Although cloud execution provides lower raw inference time, edge placement avoids Internet-backbone variability and keeps the semantic processing loop close to the audio endpoint.

VII. CONCLUSION

This paper explored the integration of AI-based semantic audio analysis within NMP systems, aiming to introduce real-time musical awareness without interfering with the low-latency audio transport path. We proposed a modular framework combining instrument recognition (PANNs), rhythmic analysis, and signal quality monitoring, and evaluated it under local, edge, and cloud deployment scenarios.

Results show that the full pipeline remains comfortably within the 2 s semantic-processing window across the evaluated execution environments. In the standalone local benchmark, the most demanding module requires approximately 28 ms and the complete pipeline approximately 38.5 ms, while the measured cloud and edge configurations exhibit higher processing costs that nevertheless remain well within the 2 s semantic-processing window. Since semantic inference runs asynchronously with respect to the JackTrip audio path, these processing times do not directly increase audio transport latency.

Beyond latency considerations, the proposed system enables meaningful semantic descriptors of the audio stream (instrument, rhythm, and quality), opening the door to adaptive mixing, intelligent effects, and context-aware network optimisation. This marks a shift from traditional semantically blind transmission toward intelligent, musically aware NMP systems.

Future work should evaluate robustness under real network conditions (e.g., 5G and LEO satellite links), extend validation beyond a single excerpt, quantify orchestration overhead in JT-aaS, and enrich the semantic layer with higher-level musical

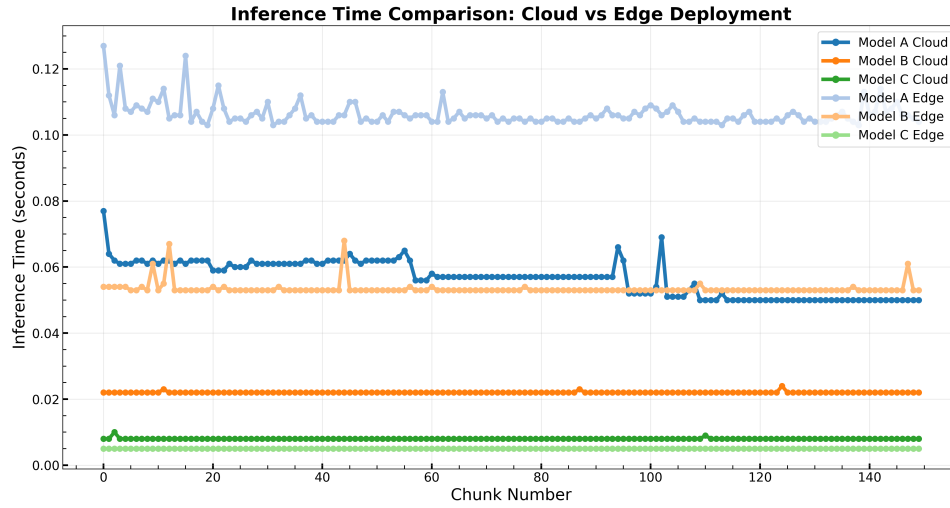


Fig. 6. Comparative analysis of inference times across consecutive chunk indices under cloud and edge deployment for all three models. Although edge deployment incurs higher raw computational latency for Models A and B, it remains well within the 2-second processing window and eliminates backbone network overhead, confirming its suitability as the primary deployment layer for semantic-aware NMP.

features such as harmony, performer identity, and expressive dynamics.

ACKNOWLEDGMENT

This work was partially supported by the Italian Ministry of Research (MUR) in the framework of the CrossLab and Forelab Projects (Departments of Excellence).

REFERENCES

- [1] C. Rottondi, C. Chafe, C. Allocchio, and A. Sarti, "An overview on networked music performance technologies," *IEEE Access*, vol. 4, pp. 8823–8843, 2016.
- [2] J.-P. Cáceres and C. Chafe, "Jacktrip: Under the hood of an engine for network audio," *Journal of New Music Research*, vol. 39, no. 3, pp. 183–187, 2010.
- [3] J.-P. Cáceres and C. Chafe, "Jacktrip/soundwire meets server farm," *Computer Music Journal*, vol. 34, no. 3, pp. 29–34, 2010.
- [4] K. Tsioutas, G. Xylomenos, and I. Doumanis, "An empirical evaluation of qome for nmp," in *2021 11th IFIP International Conference on New Technologies, Mobility and Security (NTMS)*. IEEE, 2021, pp. 1–5.
- [5] C. Drioli, C. Allocchio, and N. Buso, "Networked performances and natural interaction via lola: Low latency high quality a/v streaming system," in *International conference on information technologies for performing arts, media access, and entertainment*. Springer, 2013, pp. 240–250.
- [6] P. F. Driessen, T. E. Darcie, and B. Pillay, "The effects of network delay on tempo in musical performance," *Computer Music Journal*, vol. 35, no. 1, pp. 76–89, 2011.
- [7] E. Lakiotakis, C. Liaskos, and X. Dimitropoulos, "Improving networked music performance systems using application-network collaboration," *Concurrency and Computation: Practice and Experience*, vol. 31, no. 24, p. e4730, 2019.
- [8] R. Wilson, "Aesthetic and technical strategies for networked music performance," *AI & society*, vol. 38, no. 5, pp. 1871–1884, 2023.
- [9] L. Turchet, C. Rinaldi, C. Centofanti, L. Vignati, and C. Rottondi, "5g-enabled internet of musical things architectures for remote immersive musical practices," *IEEE Open Journal of the Communications Society*, vol. 5, pp. 4691–4709, 2024.
- [10] L. Borgianni, D. Adami, M. Bosi, S. Giordano, and C. Chafe, "A comprehensive evaluation of networked music performance using leo satellite internet: The starlink use case," *IEEE Transactions on Network and Service Management*, vol. 22, no. 5, pp. 3947–3963, 2025.
- [11] P. Verma, A. I. Mezza, C. Chafe, and C. Rottondi, "A deep learning approach for low-latency packet loss concealment of audio signals in networked music performance applications," in *2020 27th Conference of open innovations association (FRUCT)*. IEEE, 2020, pp. 268–275.
- [12] L. Vignati and L. Turchet, "On the lack of a perceptually-motivated evaluation metric for packet loss concealment in networked music performances," *Journal of the Audio Engineering Society*, 2025.
- [13] W. Bian, Y. Song, N. Gu, T. Y. Chan, T. T. Lo, T. S. Li, K. C. Wong, W. Xue, and R. A. Trillo, "Momusic: A motion-driven human-ai collaborative music composition and performing system," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 37, no. 13, 2023, pp. 16 057–16 062.
- [14] A. Boem, M. Tomasetti, L. Turchet *et al.*, "Issues and challenges of audio technologies for the musical metaverse," *J. Audio Eng. Soc.*, vol. 73, no. 3, pp. 94–114, 2025.
- [15] Q. Kong, Y. Cao, T. Iqbal, Y. Wang, W. Wang, and M. D. Plumbley, "Panns: Large-scale pretrained audio neural networks for audio pattern recognition," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 28, pp. 2880–2894, 2020.
- [16] J. F. Gemmeke, D. P. Ellis, D. Freedman, A. Jansen, W. Lawrence, R. C. Moore, M. Plakal, and M. Ritter, "Audio set: An ontology and human-labeled dataset for audio events," in *2017 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE, 2017, pp. 776–780.
- [17] B. McFee, C. Raffel, D. Liang, D. P. Ellis, M. McVicar, E. Battenberg, O. Nieto *et al.*, "librosa: Audio and music signal analysis in python." *SciPy*, vol. 2015, no. 18–24, p. 7, 2015.
- [18] J. J. Bosch, J. Janer, F. Fuhrmann, and P. Herrera, "A comparison of sound segregation techniques for predominant instrument recognition in musical audio signals," in *ISMIR*, 2012, pp. 559–564.