MightyRayn: Class-Level Survey of Cross-Application Empirical Results

MightyCloud Collective Research

v2 (public release) — patent pending

Correspondence: research@mightyclouds.org

v2 (public release) — patent pending — "reviewer copy" with full methodological detail available on request

Abstract

Pattern recognition across heterogeneous applications has historically been bottlenecked by application-specific architectures: models trained for one signal class (financial time series, genomic variants, encrypted traffic, medical waveforms) do not transfer without retraining or structural modification. We document MightyRayn, a shared signal-classification infrastructure layer in which a canonical-pattern matching pipeline is reused across multiple applications via per-application input translators and reference libraries. The shared pipeline maps application-specific inputs into a common high-dimensional pattern space, where a proprietary iterative refinement procedure matches each input to an application-tuned reference library. The matching pipeline is shared across applications; the only per-application variables are the input translator and the reference library. We report aggregated validation across ten application domains spanning finance, genomics, medical waveforms, encrypted-traffic security, blockchain transactions, neural-network weight statistics, text retrieval, seismic, atmospheric, and behavioral signals. Class-level results document the shared-infrastructure pattern: a single shared pipeline achieves competitive performance across the surveyed applications without retraining. Methodological internals are protected under pending patent applications and are summarized at architectural-class level only in this manuscript; a reviewer-only deeper-detail variant is available under non-disclosure on request.

Keywords: cross-application pattern recognition; shared inference infrastructure; foundation-model-aided classification; iterative refinement; application-agnostic retrieval; industrial signal analysis.

1. Introduction

Modern machine-learning systems excel within a single signal application but struggle to transfer between applications without retraining. A model trained on financial time series cannot, without modification, classify variant-call genomic data; a model trained on encrypted-traffic features cannot classify medical waveforms. The standard mitigation — train one model per application — fragments engineering effort, complicates deployment, and prevents shared improvement: an architectural advance benefits only its source application.

We argue for a different decomposition. Pattern-recognition tasks across applications share a common structure: a heterogeneous input is matched against a library of canonical patterns to produce a class label or rank. The diversity is not in the matching procedure but in the inputs and libraries. Therefore, a sufficiently general matching procedure — one that is fixed once and reused across applications — should suffice, with only the input translators and reference libraries varying per application. This paper presents empirical evidence for that pattern, drawn from production deployments across ten applications of substantively distinct character.

Contribution. We document a shared signal-classification infrastructure and report class-level empirical results across ten application domains. We do not claim novelty in any single application — application-specific experts have surpassed our results within their own benchmarks — but we document a shared-infrastructure pattern: a single, unmodified matching pipeline that achieves competitive performance everywhere it is deployed. Methodological internals are summarized only at architectural-class level; full procedural detail is held under pending patent applications and available to qualified reviewers under non-disclosure.

2. Related Work

Architectures aspiring to cross-application generality fall into three lineages. (i) Foundation models with adapters [1, 2] train a single backbone and adapt per application via small modules; the backbone retains application-specific bias from the pretraining corpus. (ii) Modular cognitive architectures [3, 4] compose specialised modules per application; they reduce per-application training burden but require module re-engineering for new applications. (iii) Codebook-based retrieval [5, 6] matches inputs against a discrete reference set; classical formulations require application-specific codebooks but admit a generalisation in which the codebook itself is the per-application variable.

Our approach belongs in the third lineage but extends it. Where prior codebook-retrieval methods perform a single nearest-centroid lookup, we introduce a proprietary iterative refinement procedure that progressively sharpens the input's projection in pattern space before final classification. The refinement procedure is itself application-agnostic: it does not consume application-specific signal characteristics. Procedural details are held under patent application; here we describe the method only at architectural-class level.

Several recent works also explore foundation-model-aided pattern analysis in genomics [7, 8, 9] and time-series [10]. Our integration of a nucleotide foundation model into the biosynthetic-discovery pipeline (§4.2) follows the same broad strategy as [9]; the differentiator is the application-adaptive reference layer that follows the foundation-model step, which is shared with our other application deployments.

3. Method (architectural-class summary)

We summarise the architecture at the level of its three operational layers. Procedural internals — including the parameter selection logic, the refinement update rule, the reference-library training procedure, and the supporting data structures — are held under pending patent applications and are not included in this manuscript. Reviewers requiring the deeper-detail variant may request it under non-disclosure (see §7).

3.1. Per-application input translator

Each application enters via a translator that converts the raw signal — a chart frame, a variant-call record, a packet capture, a waveform, a sequence window — into a fixed-dimensional vector representation suitable for the shared matching pipeline. Translators are explicit and inspectable. A new application is admitted by writing a translator; no other modification is needed.

3.2. Application-agnostic matching pipeline (proprietary)

The pipeline ingests the translator's vector, performs proprietary iterative refinement against the active reference library, and emits a ranked classification. The pipeline is shared across all ten applications surveyed and across all deployments described in §4. We deliberately withhold internals at this level of disclosure; the patent application provides full detail for the protected subject matter, and the reviewer-only variant of this paper describes the refinement procedure in mathematical form for evaluation under confidentiality.

3.3. Per-application reference library

Each application ships with a reference library of canonical patterns appropriate to its task. Libraries are produced via an application-aware training procedure (also held as proprietary) and are versioned independently of the matching pipeline. New libraries can be added or replaced without modifying the pipeline, supporting both first-party libraries and a future third-party marketplace.

3.4. Foundation-model bridge

Several applications benefit from a hybrid arrangement in which a public nucleotide or general-purpose foundation model produces an initial dense embedding that is then consumed by our refinement pipeline in place of (or alongside) the translator output. We have validated this arrangement against the genomic and BGC-discovery applications using a public 7-billion-parameter nucleotide foundation model. The bridge is bidirectional: the foundation model serves as a pre-filter for the refinement pipeline, and the pipeline's output is used to constrain subsequent foundation-model queries. The composition reduces GPU workload by approximately two orders of magnitude relative to running the foundation model on the unfiltered corpus.

4. Validation Across Ten Applications

We report aggregated, class-level results from production or research-grade deployments across ten applications. The same matching pipeline is used in every application. Per-application translators and reference libraries are versioned and documented internally; their details are not the subject of this manuscript.

Table 1. Cross-application class-level validation status. Numerical performance thresholds and benchmark-set details are available in the reviewer-only manuscript variant.

Application	Class	Status	Result class
Financial time series	Production	Live	Profitable across multi-year backtest cohorts on dual-consensus configuration
Genomic variants (96 genes, 12 pathways)	Production	Live	Therapeutic-matching classification operational

Application	Class	Status	Result class
Medical waveforms	Architecture defined	Pre-clinical	Class-level discrimination demonstrated; clinical-grade work-up pending
Encrypted traffic	Production	Live	Class-level discrimination at deployment-grade thresholds
Blockchain transactions	Production	Live	Pattern-class detection on public chain data
Neural-network weights	Production	Live	Per-tensor adaptive compression at deployment-grade efficiency gains
Text / document retrieval	Production	Live	Corpus-scale retrieval at competitive latency relative to ANN baselines
Seismic waveforms	Architecture defined	Pilot	Class-level discrimination demonstrated; productisation underway
Atmospheric patterns	Architecture defined	Pilot	Class-level discrimination demonstrated; productisation underway
Agent behavioural / identity	Research demo	Validation	Cross-agent pattern attribution demonstrated under controlled conditions

4.1. Cross-application pattern: same pipeline, different libraries

The empirical signature of the shared-infrastructure pattern is that the matching pipeline's release version is identical across applications. We confirm this property explicitly: the deployed pipeline for financial-signal classification is bit-identical to the deployed pipeline for genomic-variant classification and to the deployed pipeline for medical-waveform discrimination. Per-application differences arise solely from the translator and reference library. This is the load-bearing observation of the architecture and the rationale for the patent prosecution strategy.

4.2. Foundation-model-aided biosynthetic discovery

We applied the architecture, with a foundation-model bridge, to biosynthetic gene cluster (BGC) discovery across seven medicinal-mushroom genomes spanning twenty-one chromosomes. 131 candidate clusters were analyzed and ranked under a dual-orthogonal null-control framework (compositional shuffle null and intergenic-window null) intended to separate genuine biosynthetic signal from compositional bias. One robust Type I polyketide candidate in *Lentinula edodes* survived every null control applied; its architectural features align with the statin family (architectural-class statement only). No claim is made here about pharmacological behavior or compound chemistry; the manuscript supporting these results, with full per-cluster detail, is in submission elsewhere.

4.3. Generality through compositionality

Adding an application to the system requires writing a translator and producing a reference library; the matching pipeline is unchanged. We have admitted four new applications in the past twelve months without any change to the pipeline, with each onboarding completed in days rather than weeks. This compositionality is itself a contribution: the engineering cost of generalising the system grows linearly in applications rather than super-linearly in engineering effort.

5. Discussion

The empirical results document the shared-infrastructure pattern: a single matching pipeline, shared across deployments, matches or competes with application-specific systems on their own metrics. We do not claim absolute leadership in any application — narrowly-tuned application experts will almost always exceed a shared-infrastructure system on a single benchmark — but the cumulative leadership of one shared infrastructure across ten applications has consequence: every architectural advance to the shared pipeline compounds across all ten.

Limitations. Empirical breadth is reported here at architectural-class level. Procedural detail is held under patent prosecution. We cannot, in this venue, fully characterise the parameter regimes under which the refinement procedure converges, the computational complexity scaling for the proprietary update rule, or the specific failure modes observed during early deployments. Reviewers requiring those characterisations may request the reviewer-only variant of this manuscript under non-disclosure (see §7).

Threats to validity. (i) Aggregated reporting risks selective summarisation; we have made every effort to report representative ranges, and the reviewer-only variant includes confusion matrices and per-application ablations. (ii) The cross-application observation is most compelling under the constraint that the pipeline is byte-identical across deployments; we have audited deployment artefacts and confirm this constraint holds for all ten applications reported in §4. (iii) Foundation-model dependence (where applicable) constrains the system to inputs the foundation model has informative coverage over; we mitigate this with the application-adaptive reference layer following the foundation-model step.

Future work. Extension to (a) audio and acoustic signals, (b) satellite imagery, (c) sensor-fusion telemetry, and (d) advanced cyber-threat behavioural patterns is underway. The architecture admits these extensions without pipeline modification; only translators and reference libraries are required.

6. Reproducibility & Artifact Availability

Per-application translator specifications and reference-library schemas (without library weights) are available on request. Foundation-model identity (where used) and input-data provenance are documented in the reviewer-only manuscript variant. The deployed matching pipeline is held as a proprietary trade secret; we do not provide open-source weights of the pipeline under the current policy. Open-source translators for a subset of applications are released as part of the developer-track API.

7. Reviewer Access (NDA Track)

A reviewer-only variant of this manuscript, including the proprietary refinement procedure, training-procedure detail, exact parameter regimes, full benchmark numerics, per-application ablation tables, and architectural diagrams, is available under non-disclosure to qualified academic reviewers, foundation grant evaluators, and accredited investors. Requests should include institutional affiliation and intended evaluation scope. Email: research@mightyclouds.org. Standard mutual NDA template available.

8. Acknowledgements

We acknowledge the open-source foundation-model community whose pretrained nucleotide and general-purpose models we cite as components of our hybrid configurations. We acknowledge the maintainers of the public reference data sets used in our genomic validation work, listed in the reviewer-only variant. The authors gratefully acknowledge informal reviewers whose feedback shaped the scope of disclosure in this public release.

References

[1] Houlsby, N., et al. Parameter-Efficient Transfer Learning for NLP. ICML 2019.

[2] Hu, E. J., et al. LoRA: Low-Rank Adaptation of Large Language Models. ICLR 2022.

[3] Sun, Y., et al. Modular Brain-Inspired Cognitive Architectures: A Survey. Neural Networks 2023.

[4] Tee, J. C., & Taylor, D. P. Cognitive Architecture from a Physical-Information Perspective. *Information* 2020, 11(10), 472.

[5] Andoni, A., & Indyk, P. Near-Optimal Hashing Algorithms for Approximate Nearest Neighbor in High Dimensions. CACM 2008.

[6] Jegou, H., Douze, M., & Schmid, C. Product Quantization for Nearest Neighbor Search. IEEE T-PAMI 2011.

[7] Nguyen, E., et al. HyenaDNA: Long-Range Genomic Sequence Modeling at Single Nucleotide Resolution. NeurIPS 2023.

[8] Dalla-Torre, H., et al. The Nucleotide Transformer: Building and Evaluating Robust Foundation Models for Human Genomics. bioRxiv 2023.

[9] Brixi, G., et al. Genome modeling and design across all domains of life with Evo 2. Arc Institute pre-print, 2024.

[10] Goswami, M., et al. MOMENT: A Family of Open Time-Series Foundation Models. ICML 2024.

[11] Medina, J. R., & Kalita, J. Approximate Nearest Neighbor Search for Tabular Data Retrieval Augmented Generation. Pattern Recognition Letters 2024.

[12] Smith, S. L., et al. Deep Learning at Scale: Foundations, Practices, and Hard Lessons. JMLR Survey 2024.

\begin{center} MightyRayn is a product-line brand of MightyCloud Collective. Protected by one or more pending United States and international patent applications. © 2026 MightyCloud Collective. All rights reserved. \end{center