

Contracts Across Modalities: Detecting Embedding Staleness and Governance Drift in Multimodal Lakehouse Architectures

Jagan Ankam*

AI Data Engineering, IT Consulting CGI Technologies and Solutions Inc. Miami Lakes, Florida 33016, USA;

* Corresponding Author Email: jagan.ankam@gmail.com - ORCID: 0009-0006-4716-1014

Article Info:

DOI: 10.22399/ijcesn.5469

Received : 11 February 2024

Revised : 23 March 2024

Accepted : 30 March 2024

Keywords

Data contracts,
multimodal lakehouse,
vector database,
embeddings,
staleness,
provenance

Abstract:

Enterprise lakehouses increasingly colocate typed records, unstructured source objects, and vector embeddings derived from those objects. Table formats, object stores, and vector databases provide durable storage and efficient retrieval, but they do not make a derived vector self-invalidating when its source changes. An embedding can therefore remain queryable after a document is edited, redacted, or deleted; after consent is revoked; or after the active embedding model is upgraded. These failures are operationally quiet: the vector has the expected dimension, the index remains healthy, and no schema constraint necessarily fails. This paper defines a cross-modal contract as $C = (R, E, S, V, G)$, where R is a source record or object, E is its embedding, S is a provenance signature captured at embedding time, V is a validity predicate over current content and model state, and G is the governance state that authorizes retention and use. We implement the contract in a reproducible pre-production benchmark containing 180 contracts: 60 tabular records, 60 documents, and 60 images. Thirty-six are unchanged controls; 144 contain seeded failures spanning source edits, redactions, deletions, model-version upgrades, consent revocations, vector corruption, and unversioned model-artifact replacement. Lightweight signature, model-version, and governance checks detect 120 of 144 failures (83.3%) with no false positives. They intentionally miss vector corruption and unversioned artifact replacement because neither changes the stored source signature, declared version, or governance metadata. Selective re-embedding detects those residual classes; the combined policy detects all 144 seeded failures in this benchmark. Median lightweight evaluation ranges from 6.5 to 27.8 μ s per contract, whereas re-embedding ranges from 0.42 to 2.84 ms, a 62-102 cost difference. The contribution is not a claim that hashing solves semantic quality. It is a precise separation of three obligations that current systems often conflate: source freshness, model compatibility, and governance authority, together with an escalation path for failures that require inspection of the vector payload itself.

1. Introduction

The enterprise data platform is no longer organized around tables alone. A modern analytical or retrieval workload may use a typed customer row, the PDF or image from which part of that row was extracted, and one or more vector embeddings used to retrieve semantically related objects.

These artifacts are often stored in different physical systems but presented as a unified lakehouse: table metadata and snapshots in an open table layer, raw files in object storage, vectors in a specialized index, and governance metadata in a catalog. This arrangement is operationally convenient because it supports business intelligence, machine learning,

and retrieval-augmented generation without forcing every modality into a single storage representation.

The unification is incomplete, however. A schema contract can state that a field is a non-null decimal, that a key is unique, or that a timestamp uses a declared type. The analogous obligation for an embedding is harder to state. A vector is not a stable fact copied from a source. It is a lossy result of content extraction, chunking, preprocessing, model parameters, and numerical execution. Its validity is relational: the vector is useful only while it remains bound to the same source state, generated by an allowed model and preprocessing configuration, and retained under an active governance basis.

This relational validity is easy to break without producing a conventional failure. A document can be corrected while the old vector remains in an approximate nearest-neighbor index. A customer can revoke consent while the derived embedding remains retrievable. A privacy workflow can delete a row from a governed table but omit a separate vector collection. A team can deploy a new embedding model while retaining old vectors under the same logical index name. In each case, the index can continue to return results with normal latency and healthy infrastructure metrics. The defect is not an exception; it is a broken relationship among artifacts.

The distinction matters because provenance and governance do not end at the source boundary. Classical database provenance asks why an output exists and where its contributing values originated [3]–[5]. Modern lineage systems capture workflow execution and dependency graphs [6]–[8]. Responsible- AI literature likewise emphasizes documentation of datasets, models, and derivation processes [21]–[23], [27]. Yet most operational controls stop short of making a derived embedding invalid when the content, model, or legal authority that justified it changes.

This paper addresses that gap with a deliberately small formal core and an executable benchmark. We define a cross-modal contract $C = (R, E, S, V, G)$. The provenance signature S' records the canonical content state of R at embedding time. The predicate V requires the current source signature and declared active model version to match the embedding record. The governance component G records consent, PII classification, retention, redaction, and deletion state. A lightweight evaluator checks these fields without recomputing every embedding. When the lightweight evidence is insufficient—for example, the vector payload is corrupted or a model artifact changes without a version increment—a deep verifier re-embeds the source and compares the result with the stored vector.

The empirical study uses real executable embedding pipelines rather than simulated detector scores. Textual records and documents are embedded with locally fitted TF-IDF/SVD pipelines; images are embedded with histogram-of-oriented-gradients and principal-component pipelines. These models are intentionally deterministic, versionable, and inexpensive. They are not presented as state-of-the-art semantic encoders. Their purpose is to exercise the lifecycle and governance properties of derived vectors under controlled model changes. The framework is model-agnostic and can be attached to transformer or multimodal encoders in production. Figure 1 summarizes the problem. The source, embedding

pipeline, vector, and index can all remain individually wellformed while the relationship among them becomes invalid.

A. Problem Statement

Existing data-contract mechanisms are strongest where the governed object has stable, declarative structure. They can reject a missing column, an incompatible type, or a violated nullability rule. An embedding does not expose its correctness through those properties. Dimension and numeric type establish only that a vector can be stored and compared; they do not establish that it represents the current source, that it was produced by the approved model, or that the organization still has authority to retain and query it.

The resulting compliance and reliability problem has four parts. First, source staleness: the source object changes while its embedding is not regenerated. Second, model drift: the approved embedding model or preprocessing pipeline changes while old vectors remain mixed with new ones. Third, governance drift: deletion, redaction, retention, or consent actions applied to the source do not propagate to the derived vector. Fourth, payload drift: the embedding itself is corrupted or produced by an undeclared artifact even though metadata claims that nothing changed. Current storage unification does not, by itself, specify which of these conditions invalidates a vector or which enforcement action must follow. This silent divergence parallels operational drift in production database systems, where nominally healthy infrastructure can conceal an accumulating gap between actual and governed state [37].

The practical question is therefore not merely how to store tables, objects, and vectors together. It is how to make the relationship among them testable and enforceable at acceptable cost. A useful mechanism must detect common invalidations without re-embedding an entire corpus after every change, preserve enough evidence for audit, avoid false alarms on unchanged objects, and acknowledge the classes of failure that metadata checks cannot see.

2. Research Gap and Contributions

Three adjacent research areas do not fully cover this problem. Data-quality and contract systems focus primarily on observable properties of records and datasets [1], [2], [34], [35]. Provenance and lineage systems describe derivation and execution history, but lineage alone does not necessarily evaluate whether a derived representation is still current or authorized [3]–[8]. Vector database research focuses on indexing, approximate search, hybrid query processing, and storage efficiency

[11]– [14]; it typically assumes that the vectors presented to the index are valid application data. The missing abstraction is an enforceable contract that crosses representation boundaries. Such a contract must bind a vector to (i) a specific source state, (ii) a declared model state, and (iii) a current governance state. It must also distinguish cheap invalidity evidence from failures that require recomputation.

This paper makes five contributions:

- 1) It formalizes a cross-modal contract $C = (R, E, S, V, G)$ and defines validity, governance alignment, and payload integrity as separate obligations.
- 2) **It specifies a two-tier enforcement architecture:** lightweight signature/version/governance evaluation followed by selective deep re-embedding when metadata evidence is insufficient.
- 3) It provides a reproducible pre-production benchmark of 180 tabular, document, and image contracts with 144 seeded failures and 36 controls.
- 4) It reports task-level detection, false-positive, threshold sensitivity, and latency results from executable local embedding models.
- 5) **It translates the formalism into deployment actions:** block retrieval, quarantine, regenerate, tombstone, invalidate an ANN entry, and record auditable evidence.

The study is intentionally bounded. It does not claim that content hashing verifies semantic correctness, that a synthetic corpus represents all enterprise data, or that the measured latency predicts a specific commercial platform. The contribution is the contract boundary and the empirical separation between what can be established cheaply from metadata and what requires a deep vector check.

3. Related Works

A. Data Quality and Data Contracts

Data-quality research distinguishes intrinsic accuracy from contextual fitness for use and emphasizes that validity depends on the consumer and task [1], [2]. Large-scale verification systems operationalize these ideas through declarative constraints, statistical profiles, and continuous validation [34], [35]. These mechanisms are necessary in a multimodal lakehouse because embeddings often inherit identifiers, partitions, and access policies from structured records. They are not sufficient because a vector can satisfy dimensional and type constraints while representing obsolete content.

The contract proposed here extends rather than replaces ordinary data contracts. A table contract protects the shape and selected semantics of R . The cross-modal contract protects the derivation edge $R \rightarrow E$. This distinction matters during partial updates. A document row may remain structurally valid after its body is edited, yet the old vector becomes stale. Conversely, a model-version change can invalidate E even when R is unchanged.

B. Provenance, Lineage, and Reproducibility

Database provenance provides the conceptual foundation for binding outputs to inputs. Buneman et al. distinguish why- and where-provenance [4]; Cheney et al. survey why, how, and where provenance and their uses in debugging and update propagation [5]. The Open Provenance Model formalizes entities, processes, and agents [6]. Workflow and data-science systems record execution context to support lineage and reproducibility [7], [8].

Our provenance signature is intentionally narrower than a complete lineage graph. S is a compact content-state witness used for fast invalidation. It does not replace detailed lineage describing extraction, chunking, feature generation, and indexing. In a production implementation, the contract registry should link to that lineage. The narrow signature is valuable because it supports an $O(1)$ equality test for the common question: is this embedding still bound to the current canonical content?

C. Lakehouse and Vector Storage Systems

Transactional lakehouse formats provide ACID commits, schema evolution, snapshot isolation, and time travel over object storage [9]; lakehouse architecture research places such capabilities within integrated warehouse-and-lake designs [10]. These properties make source state addressable and can provide stable snapshot identifiers for R . Vector database systems and ANN benchmarking research provide efficient storage, retrieval, and comparative evaluation of high-dimensional representations [11]–[14]. Their performance work addresses indexing structures, quantization, hardware acceleration, and hybrid predicates.

These systems solve different parts of the stack. A table snapshot can show that a source changed; a vector index can efficiently query E . Neither fact alone ensures that a source update invalidates the corresponding vector entry. The proposed registry supplies that missing binding and makes enforcement independent of whether the physical platforms are a single product or several services.

D. Embedding Models and Model Lifecycle

Sentence and image encoders map variable-sized inputs into fixed-dimensional vectors [15]–[18].

Embeddings are model-dependent: changing tokenization, feature extraction, parameters, normalization, or training data can alter the geometry of the resulting space. Mixing vectors generated by incompatible models can therefore make distance comparisons meaningless even when all vectors have the same length.

Production-ML research has long identified operational pitfalls, hidden dependencies, and readiness debt as major lifecycle risks [24], [25]. Model cards and dataset documentation improve transparency about intended use, evaluation, and provenance [21]–[23]. The contract in this paper operationalizes one small part of that documentation: the embedding record carries a model identifier and version, and the validity predicate compares it with the currently approved model state.

E. Governance, Erasure, and Derived Artifacts

Data governance becomes difficult when information is transformed into derived artifacts. Provenance has been proposed as a foundation for responsible AI because it connects outputs to origins and processing history [27]. Privacy research shows that learned representations and models may retain information about source data and can be subject to inversion or membership attacks [28], [31], [32]. Machine-unlearning work addresses the harder problem of removing influence from trained models [29], [30].

An embedding index is simpler than a trained foundation model in one respect: a vector is usually an individually addressable derived record. Deleting or regenerating that record is feasible if the system knows which source and governance state it belongs to. The failure is frequently orchestration rather than theoretical impossibility. The proposed G component makes consent, deletion, redaction, and retention changes part of the same contract evaluation used for freshness.

4. Formal Cross-modal contract framework

A. Contract Definition

Let R denote a source artifact. R may be a canonical serialization of a tabular row, the bytes or extracted text of a document, or the bytes of an image. Let $M_{:,v}$ denote an embedding pipeline with model identifier 7 and version v , including preprocessing that materially affects the representation. The derived embedding is

$E = M_{:,v}(R)$. (1) The cross-modal contract is $C = (R, E, S, V, G)$, (2)

where $S = H(k(R))$ is a cryptographic digest of the canonicalized source $K(f)$ at embedding time; V is the validity predicate; and G is the governance

state captured and compared across the derivation edge.

In the implemented benchmark,

$V_{\text{light}} = I[S = H(K(R_z))] \wedge I[v = 4\%] \wedge AL[G = G_i], @B)$

where subscript t denotes the current state. A missing source makes the signature condition false and, when deletion is authorized, requires the derived vector to be tombstoned or removed. The lightweight contract is valid only when all three components match.

The deep payload predicate is

$V_{\text{deep}} = I[\cos(E, M_{i,v}(R_t)) > 7]$, (4)

for a configured threshold 7 . The complete benchmark policy accepts a vector when V_{light} holds and no deep verification is required, or when a triggered deep verification also satisfies

V_{deep} . Operationally, the trigger can be periodic sampling, unexplained retrieval degradation, index corruption evidence, or a high-risk asset classification.

B. Semantics of the Five Components

Table I gives the enforceable meaning of each component. R is not necessarily stored inside the registry; a stable URI and source key are sufficient if the evaluator can resolve current content. E may be stored in a vector database or sidecar index. S is a compact witness, not a semantic checksum. V is executable policy rather than descriptive metadata. G' captures the authority to retain and use the derived representation.

C. Canonicalization and Signature Scope

A hash is meaningful only relative to canonicalization. For structured rows, field ordering, null representation, numeric formatting, and ignored operational fields must be fixed. For documents, line endings, extraction version, and text normalization must be declared. For images, the contract can hash raw bytes, decoded pixels, or a normalized rendition; each choice detects different changes. The benchmark hashes canonical UTF-8 text for records and documents and raw PNG bytes for images.

A production design should include the canonicalization or preprocessing fingerprint in the model identity. Otherwise, an extraction library can change output while the nominal embedding model version remains constant. The benchmark includes an intentionally unversioned model-artifact change to demonstrate this blind spot.

D. Governance Propagation

Governance state is not merely an annotation copied for search filtering. It is part of the vector's validity. Let G include consent c , deletion d , redaction r , retention deadline q , and PII class p . A simple authorization predicate is

(5) with domain-specific exceptions for lawful basis and retention holds. If $A(G;)$ becomes false, retrieval must be blocked before asynchronous cleanup completes. This distinction between authorization and physical deletion prevents a race in which a vector remains searchable during a downstream deletion job.

$A(G_i) = A_{ad}; A_{now} < q@,$

E. Enforcement Architecture

Figure 3 shows the proposed architecture. Source changes arrive through table change-data capture, object notifications, or governance/IAM events. The evaluator resolves the contract registry, recomputes the signature when required, compares model state and governance, and produces a policy decision. High-confidence invalidity triggers deterministic actions. Ambiguous or payload-level risks are routed to selective deep verification.

The policy engine should emit idempotent commands. A deletion event may be delivered more than once; repeated tombstone or delete operations must not resurrect the vector or fail the workflow. Audit records should include the source key, prior and current signatures, prior and current model versions, governance transition, action, timestamp, and outcome.

5. Research Questions and hypotheses

The benchmark addresses four research questions.

RQ1: How accurately do signature, model-version, and governance checks detect seeded cross-modal contract failures?

RQ2: What failure classes remain invisible to the lightweight predicate, and can selective re-embedding detect them?

RQ3: Does detection performance differ across tabular, document, and image modalities?

RQ4: What is the observed cost difference between lightweight evaluation and deep re-embedding in the controlled environment?

The corresponding hypotheses are:

H1: The lightweight union will detect all source-state, declared model-version, and governance-state failures without flagging unchanged controls.

H2: The lightweight union will not detect vector corruption or an unversioned model-artifact replacement because its observed metadata remains unchanged.

H3: Adding selective re-embedding will increase recall for the residual payload-level failures.

H4: Lightweight evaluation will be at least one order of magnitude faster than re-embedding for every modality in this benchmark.

6. Methodology

A. Study Design

We use a controlled pre-production benchmark rather than production telemetry. This choice allows the ground truth to be known: each contract has a declared scenario, and unchanged controls provide a direct false-positive measurement. The methodology follows controlled software-engineering experimentation principles by fixing the random seed, documenting the artifact generation procedure, isolating detector outputs, and releasing task-level results [36]. The unit of analysis is one cross-modal contract, not one query. The experiment asks whether a stored vector should remain valid after the corresponding current state is constructed. Retrieval quality is outside scope; the benchmark evaluates lifecycle integrity.

B. Corpus Construction

The corpus contains 180 contracts evenly divided among three modalities:

- Tabular: canonical JSON serializations of customer-like records with identifiers, region, segment, annual value, consent, and risk tier.

- Document: short synthetic operational notes covering claims, onboarding, network operations, due diligence, fraud monitoring, and reporting.

- Image: generated 128x128 PNG objects containing colored geometric forms and stable identifiers.

No production, personal, or externally sourced sensitive data are used. Each modality contains 60 contracts: 12 controls, eight each for five primary drift scenarios, and four each for two payload-level scenarios.

C. Experimental Embedding Models

The experiment uses deterministic local models so every artifact can be regenerated without an external API. This is a methodological decision, not a claim that classic pipelines represent the accuracy of current foundation models.

For tabular and document content, version 1.0 uses word uni- and bi-gram TF-IDF with 512 features followed by 48-component truncated SVD and L₂ normalization. Version 2.0 uses character 3–5-grams with 768 features and a separately fitted 48-component SVD. The unversioned replacement (variant 1b) uses binary word 1–3-grams and a different SVD fit while retaining the declared version string 1.0. Truncated matrix factorization provides a compact dense representation and deterministic transformation under a fixed fitted artifact [19]. For images, version 1.0 concatenates HOG features (nine orientations, 16-pixel cells) with RGB histograms and reduces them to 32 dimensions by PCA. Version 2.0 changes orientations, cell size, histogram resolution, and

PCA fit. Variant 1b changes the HOG configuration and uses a grayscale histogram while retaining version 1.0. HOG provides a real contentsensitive visual descriptor [18]; PCA provides a fitted dense embedding space [20].

The model inventory in Table III makes a point often missed in embedding governance: “model version” must cover the complete representation pipeline. Changing tokenization, extraction, feature configuration, normalization, or dimensionality reduction can invalidate comparability even when the branded encoder name is unchanged.

Transformer encoders such as BERT and Sentence-BERT and convolutional encoders such as ResNet are natural production extensions [15]-[17]. They were not downloaded or called in this controlled environment. Substituting them changes embedding cost and semantic behavior, but not the signature, version, or governance logic being evaluated.

D. Detectors

Six detector configurations are reported.

- 1) **Signature:** Flags missing current content or a SHA-256 mismatch.
- 2) **Model tag:** Flags a declared current-version mismatch.
- 3) **Governance:** Flags a change in the governance tuple or a deleted source.
- 4) **Lightweight union:** Logical OR of the first three detectors.
- 5) **Deep re-embedding:** Recomputes the embedding with the active artifact and flags cosine similarity below $\tau = 0.999$.
- 6) **Combined:** Logical OR of lightweight and deep evidence.

The deep detector is unavailable for a deleted source because there is no object to re-embed. This is precisely why governance and source existence cannot be delegated to vector similarity.

E. Metrics

For each detector we report true positives (TP), false negatives (FN), true negatives (TN), false positives (FP), recall, specificity, precision, and accuracy. Recall is $TP/(P+ FN)$; specificity is $TN/(I.N + FP)$. The 36 controls are used for the false-positive denominator. We also report scenario-level similarity distributions, and threshold sensitivity. Timing uses `perf_counter_ns` around the executed check on each contract. The lightweight time includes signature recomputation, version comparison, and governance comparison. Deep time includes feature extraction and embedding transformation for one current source. File and model initialization are excluded because they are amortized services in a deployed evaluator.

7. Experimental Setup

The benchmark was executed with Python 3.13.5 on Linux 6.12 using five virtual Intel Xeon Platinum 8573C CPU cores and 5.9 GiB memory. The principal libraries were NumPy 2.3.5, pandas 2.2.3, scikit-learn 1.8.0, Pillow 12.2.0, and scikit-image 0.26.0. The deterministic seed was 20260728. All embeddings were generated locally on CPU. The stored contract registry contains the source URI, modality, model identifier, model version, source signature, governance snapshot, and embedding dimension. Vectors are released in a separate CSV. Source artifacts are released as canonical record/document content and generated PNG files. The benchmark script regenerates the corpus, fits all model variants, seeds current-state failures, runs the detectors, and exports task-level results. No ANN index is needed to test validity because the contract check is keyed by source/vector identity rather than nearestneighbor search. In a deployed system, the policy engine would use the vector-store primary key to tombstone or delete the corresponding index entry. Approximate indexing research remains relevant to the cost and mechanics of that operation {11J}-{14}.

8. Results

A. Aggregate Detection Performance

Table IV reports aggregate results over 144 seeded failures and 36 controls. Signature checking detects the 72 content-state failures: source edits, redactions, and deletions. Model checking detects the 24 declared upgrades. Governance checking detects 72 failures because redaction, deletion, and consent transitions alter governance or source authority. These component counts overlap; they must not be summed as independent coverage. The lightweight union detects 120 of 144 failures, for 83.3% recall, and accepts all 36 controls. Its 24 false negatives are exactly the two scenarios designed to preserve the observed metadata: 12 corrupted vectors and 12 unversioned modelartifact changes. H1 and H2 are therefore supported within the benchmark. Deep re-embedding detects 96 failures. It identifies source edits, source redactions, declared model upgrades, vector corruption, and unversioned artifact replacement. It does not detect consent revocation because the content and model output are unchanged, and it cannot run on deleted sources. This result is important: re-embedding is not a substitute for governance evaluation. The combined policy detects all 144 seeded failures with no false positives, supporting H3 for the tested scenarios.

The perfect specificity should not be generalized. Controls were generated with deterministic canonicalization and embedding execution. Real systems can produce benign byte changes, nondeterministic model output, floating-point differences, or delayed event ordering. Those conditions require canonicalization rules, temporal grace periods, and calibrated deep thresholds.

B. Scenario-Level Results

Figure 5 shows the detector-by-scenario matrix. The pattern validates the intended division of responsibility.

Source edits are detected by both the signature and deep checks. The signature result is decisive and cheaper; deep similarity adds no governance evidence. Redactions likewise change content and governance. Treating redaction only as a metadata filter would be unsafe because the old vector can still encode the removed material. Deletions are detected by source existence and governance, while re-embedding is impossible. Model upgrades are detected immediately by version mismatch; deep comparison confirms that the spaces differ but should not be needed to establish invalidity. Consent revocations are detected only by governance because the content and embedding remain numerically identical. Vector corruption and unversioned artifact replacement require deep verification because the lightweight evidence was intentionally held constant.

This decomposition answers RQ2. The residual failures are not random misses; they are failures outside the observation model of the lightweight predicate. A hash over the source cannot prove that the stored vector bytes are intact. A modelversion string cannot prove that the deployed artifact behind that string was not replaced. The correct response is to extend the evidence model—for example with an embedding checksum and signed model-artifact digest—or to schedule deep verification for risk-selected assets.

C. Results by Modality

Because each scenario is evenly allocated across modalities, the lightweight detector achieves the same scenario recall for tabular, document, and image contracts. The execution cost

D. Latency and Cost

Median lightweight evaluation is 6.48 μ s for documents, 6.83 μ s for tabular records, and 27.81 μ s for images. Median deep re-embedding is 0.432 ms, 0.424 ms, and 2.839 ms respectively. The deep-to-lightweight median ratio is 66.6x for documents, 62.0 for tabular records, and 102.1x for images. H4 is supported.

Figure 6 plots the observed coverage-cost frontier using the conservative image latency for the deep

configurations. The result supports a tiered policy. Cheap checks should run on every relevant event or read path. Deep verification should be targeted, sampled, or invoked when higher-assurance evidence is required.

The absolute timings should not be compared directly with transformer APIs or GPU inference. The ratio is the more durable observation: comparing hashes and metadata is qualitatively cheaper than regenerating model output. With large documents, image encoders, remote inference, rate limits, and egress, the production gap is likely larger, but that is an inference rather than a measured result here.

E. Threshold Sensitivity

The selected deep threshold $\tau = 0.999$ detects all 24 residual payload failures and produces no false positives among controls. At $\tau = 0.995$, residual recall falls to 87.5%; at 0.99 it falls to 83.3%; at 0.95 it falls to 45.8%. Controls remain at cosine similarity 1.0 because execution is deterministic.

This sensitivity reveals why a universal cosine threshold is dangerous. A threshold suitable for deterministic local pipelines may be too strict for nondeterministic serving or numerically different hardware. Conversely, a looser threshold can miss small but real corruption. Production calibration should use repeated clean runs, per-model distributions, and applicationspecific risk rather than copying 0.999 from this experiment.

F. Governance Propagation Results

The benchmark contains 72 governance-relevant transitions: 24 redactions, 24 deletions, and 24 consent revocations. Before enforcement, every scenario intentionally retains the old embedding. The lightweight contract alerts on all 72. Redaction sources remain available and can be regenerated or deleted. Deleted sources cannot be re-embedded and require vector deletion or tombstoning. Consent revocations leave the source technically available but remove the authority to use the vector; the correct action is access denial followed by policy-defined deletion.

The distinction between detection and propagation is critical. Detecting G # G; does not itself remove a vector. The architecture must couple the alert to an idempotent action and verify completion. A compliance claim should therefore report at least three timestamps: governance event received, retrieval blocked, and physical deletion or regeneration completed.

9. Discussions

A. What the Results Establish

The experiment establishes that a small amount of explicit metadata can detect broad classes of silent embedding invalidity. The result is not surprising in isolation: a content hash detects content changes, and a version comparison detects version changes. The contribution is the contract that treats these facts and governance state as one release decision across modalities, plus the empirical evidence that the union has a clear boundary.

The boundary is as important as the coverage. An organization that stores only a source hash and model version may believe it has solved freshness. It has not established vector integrity, model-artifact identity, or governance propagation. The benchmark makes those missing obligations visible through controlled false negatives.

B. Why Schema and Index Health Are Weak Proxies

Every seeded vector remains a valid numeric array of the expected dimension. Approximate-search infrastructure can index and return it. A schema contract over the vector column therefore passes. Index health also passes because graph connectivity, segment availability, and query latency do not reveal semantic linkage. These controls answer whether the vector can be served, not whether it should be served.

A robust observability design should keep these layers separate. Storage health protects availability. Vector-distribution monitoring can reveal large geometric shifts. Cross-modal contracts protect item-level provenance and authority. Retrieval evaluation measures application quality. No single metric substitutes for the others.

C. Contract Granularity

The benchmark creates one contract per source object or row. Production document pipelines often chunk a source into many embeddings. In that case, R should identify both the source snapshot and the chunk definition. A source edit may invalidate all chunks or only those overlapping the changed region. Fine-grained chunk signatures reduce regeneration cost but complicate lineage and deletion completeness.

For images, multiple representations may coexist: thumbnail, OCR text, caption, visual embedding, and region embeddings. Each derived artifact should have its own FE, S, and model lineage while inheriting governance from the source. A contract registry can represent this as a derivation graph rather than a single tuple per object.

D. Model-Version Semantics

The unversioned model-change scenario exposes a common governance weakness. Teams frequently version the named encoder but not the full artifact: tokenizer revision, extraction code, normalization,

quantization, prompt template, or feature configuration. The contract should therefore use a signed pipeline fingerprint, not a manually typed label alone. A practical fingerprint can hash the model artifact digest, preprocessing package lockfile, canonicalization specification, and relevant runtime configuration. Version mismatch is a validity decision, not necessarily an immediate requirement to regenerate every historical vector. Some applications can operate separate indexes per model generation and retire old versions gradually. What is unsafe is comparing vectors from incompatible spaces without an explicit compatibility policy.

E. Governance Is a Read-Path Concern

Many deletion architectures are asynchronous. That is operationally reasonable, but an asynchronous cleanup queue should not leave the vector readable. The contract evaluator can enforce a fast logical block using current governance state while background jobs delete physical copies and rebuild indexes. This pattern mirrors database tombstones and access-control revocation: deny first, compact later.

The same rule applies to consent and retention. A vector's presence in an index does not establish ongoing authority. Retrieval should evaluate or materialize an authorization state tied to G,. Caches and replicated indexes must receive the same invalidation event, and completion must be audited.

F. Relationship to Machine Unlearning

Deleting an individually stored embedding is not equivalent to machine unlearning. If the source was used to train or finetune a model, its influence may persist in model parameters, and removal requires different techniques [29], [30]. This paper addresses derived retrieval artifacts that remain individually addressable. A complete governance program must trace both branches: source-to-vector derivation and source-to-trained-model influence.

10. Practical Implications

A. Minimum Registry Schema

A deployable registry should contain: source identifier and modality; source snapshot or object version; canonicalization version; source signature; vector-store key; embedding dimension; model identifier; model artifact digest; model and preprocessing version; creation timestamp; consent and legalbasis state; PII class; retention deadline; redaction/deletion state; last validation time; and enforcement status. The registry should be transactional with either the vector write or an

outbox event so that orphaned vectors and orphaned contracts are detectable.

B. Event-Driven Evaluation

Full scans are useful for reconciliation but should not be the only control. Source updates, object overwrites, table deletes, model deployments, consent changes, and retention expiration are natural triggers. Each event resolves affected contract keys and runs the lightweight predicate. Batch systems can evaluate changed partitions; streaming systems can evaluate individual keys.

Out-of-order events require temporal logic. A vector generated after a source edit should not be invalidated by an older delayed event. Registry updates should compare source snapshot sequence, event time, or transaction version. Idempotency keys prevent duplicate deletions or refreshes.

C. Risk-Tiered Deep Verification

Deep checks should be allocated according to impact and uncertainty. High-risk candidates include regulated documents, consent-sensitive customer content, high-value retrieval corpora, unexplained embedding-distribution changes, model deployments without artifact attestation, and random samples used to estimate silent corruption. Lower-risk stable content can rely primarily on event-driven lightweight evaluation plus periodic sampling.

A useful operational metric is the deep-verification yield: the proportion of sampled vectors that fail V_{deep} despite passing V_{light} . A rising yield indicates missing versioning, storage corruption, or unmodeled preprocessing changes and should trigger broader investigation.

D. Release and Audit Gates

The proposed framework can be implemented as four gates: 1) Write gate: do not publish E without a complete registry record and source signature. 2) Change gate: source/model/governance events evaluate affected contracts and block invalid vectors. 3) Read gate: retrieval filters out contracts in invalid, quarantined, or deletion-pending states. 4) Audit gate: reconcile sources, registry, vector store, and deletion evidence on a schedule. The read gate is essential. A change detector that merely opens a ticket does not close a compliance gap. Enforcement status must be visible to retrieval infrastructure.

11. Threats to Validity

A. Construct Validity

The benchmark operationalizes staleness as canonical content mismatch and model drift as declared version mismatch. These are precise but

incomplete constructs. Semantic relevance can change without a byte change, and a byte change can be semantically irrelevant. Governance rules are simplified to a small tuple and do not model every lawful basis, litigation hold, or jurisdictional exception.

Cosine re-embedding is used as a payload check, but similarity thresholds are model-specific. It detects that vectors differ; it does not prove which vector is correct or whether retrieval quality is materially affected.

B. Internal Validity

Scenario allocation is deterministic and balanced. This improves reproducibility but may make the corpus easier than uncontrolled production data. The same implementation generates the source and evaluates the hash, reducing integration errors that would exist across independent services. Timing measurements include warm local model objects and exclude network, queue, catalog, and vector-store latency.

The deep detector uses deterministic pipelines. Nondeterministic encoders could create false positives under a strict threshold. Production systems should repeat clean embeddings and measure hardware and serving variance before selecting T.

C. External Validity

The corpus contains 180 synthetic contracts and simple objects. It does not represent long PDFs, scanned forms, videos, audio, multilingual content, billion-vector indexes, or complex chunk graphs. The measured recall applies to the seeded scenarios, not to all possible embedding failures. Commercial platforms may provide native synchronization or governance features that change the integration burden.

The local LSA and HOG/PCA models are not representative of the latency or semantic behavior of transformer and multimodal encoders. The contract logic is model-agnostic, but cost ratios and threshold distributions must be remeasured for each deployment.

D. Conclusion Validity

There are only 36 controls, so the absence of false positives does not establish a population specificity of 100%. Scenario counts are fixed rather than sampled from an industry distribution; aggregate accuracy therefore depends on benchmark composition. The strongest conclusions are mechanism-level: which fields can observe which changes, and which failures remain unobservable without additional evidence.

12. Future Work

The next evaluation should replace the local models with at least two text encoders and two image or multimodal encoders, repeat each embedding across hardware and serving configurations, and calibrate model-specific similarity thresholds. A larger corpus should include chunked PDFs, OCR revisions, image transcoding, duplicate objects, multilingual content, and schema-linked records.

The formal contract should be extended with a preprocessing fingerprint P and model artifact digest A , yielding $(R, E, 8, P, A, V, G)$. This would make the unversioned artifact failure cheaply detectable. An embedding checksum can detect storage corruption without re-embedding, although it cannot prove semantic linkage to the source.

Further work should also evaluate end-to-end governance propagation: time from consent or deletion event to read blocking, physical deletion, ANN index compaction, cache invalidation, and audit completion. Fault injection across queues and partial service outages would test whether the architecture fails closed.

Finally, the benchmark should measure application impact. Retrieval experiments can quantify how stale or unauthorized embeddings change nearest-neighbor results, RAG citations, and downstream decisions. That extension would connect contract validity to user-visible harm rather than treating all invalid vectors as equally consequential.

13. Conclusion

Multimodal lakehouses create a relationship that ordinary schema contracts do not govern: a source artifact, a model-dependent embedding, and the authority to retain and use that embedding. The relationship can break silently while every component remains individually well-formed.

This paper defined an enforceable cross-modal contract $C = (R, E, S, V, G)$ and implemented it in a reproducible 180-contract benchmark. Lightweight source-signature, modelversion, and governance checks detected 120 of 144 seeded failures with no false positives in the controlled corpus. The 24 misses were not unexplained noise; they were vector corruption and an unversioned model-artifact replacement, both outside the lightweight observation model. Selective re-embedding detected those residual classes, while governance checks remained necessary for deletion and consent events that vector similarity cannot observe. The combined benchmark policy detected all seeded failures.

The practical conclusion is disciplined rather than absolute. Hashes do not prove semantic quality, and re-embedding does not prove governance authority. The two checks answer different questions. A

defensible architecture binds source, vector, model, and governance evidence in one registry; runs cheap evaluation on every relevant change; blocks invalid vectors on the read path; and reserves expensive verification for payload-level uncertainty. That is the minimum needed to turn embedding freshness and governance propagation from informal assumptions into testable system behavior.

DATA AND ARTIFACT AVAILABILITY

The supplementary package contains the benchmark generator and evaluator, 180 synthetic source artifacts, contract registry, embedding vectors, task-level outcomes, aggregate metrics, latency and threshold-sensitivity tables, figures, and a data dictionary. The corpus contains no production or personal data. Results can be regenerated by running `code/run_benchmark.py` followed by `code/make_figures.py`.

Appendix A Formal Properties and Observation Boundaries

A. Observed and Unobserved State

The lightweight evaluator is useful only if its observation boundary is explicit. Let the complete operational state of a contract at time t be

$$X_t = (R_i, E_x, M_i, G_t, A_r), (6)$$

where MM ; is the executable model artifact, including preprocessing, and A , denotes auxiliary storage state such as vector bytes, index membership, and tombstones. The lightweight evaluator does not observe all of X ; It observes

$$O; = (H(R_i), m_e, G_t, @2), (7)$$

where $H(R_i)$ is the current canonical source signature, m ; is the declared model version, and x ; is source existence. Its stored evidence is $O, = (S, m_-, G_-, 1)$ at embedding time. The predicate can therefore be written as

$$V_i(C, t) = [H(R_z) = S] \wedge [m_e = m_e] \wedge AT(G_i, G_e)$$

A x_e , (8) where I is the governance-compatibility policy. A deep verifier adds an observation of the current model output $F_{yy}(R,)$ and compares it with F ;

Proposition 1 (completeness within the observation boundary): If a seeded failure changes at least one field tested by V_i , and canonicalization and event ordering are correct, the lightweight evaluator detects that failure.

Justification: Each conjunct is an equality or policy test over a stored/current pair. A source edit, redaction, or deletion changes the source signature or existence. A declared model upgrade changes m .. A consent, retention, or deletion transition changes G ; and therefore I '. No similarity threshold is required for these decisions.

Proposition 2 (indistinguishability outside the boundary): If two operational states X_i and X_j induce the same observed state O , but differ in an unobserved component that changes embedding validity, no detector using only O ; can distinguish them.

Justification: The detector receives identical inputs in both states. Vector corruption changes FE , but not $H(Rz)$, $m+$, or G ;. An artifact replacement can change M , while leaving the declared version m ; unchanged. These are not implementation bugs in the equality checks; they are information-theoretic blind spots in the selected evidence model.

This distinction prevents an inflated claim. The benchmark's 83.3% lightweight recall is not a generic ceiling for hashes and metadata. It is the consequence of the seeded scenario mixture: 120 failures modify observed evidence and 24 modify only unobserved state. Adding a vector checksum would move accidental payload corruption into the cheap observation boundary. Adding a signed digest of the complete model artifact would do the same for undeclared artifact replacement. Deep re-embedding is one way to compensate when those fields are unavailable, but it is not the only possible extension.

B. Validity, Authorization, and Physical State

A single Boolean named "valid" can hide three different obligations. We separate them as

$V_p = [H(Rz) = S] \wedge [m = m]$, (9) $V_a = T(G_i, G_-)$, (10) $V_p = (Et, At)$, (89)

where V_p is freshness/compatibility, V_a is authorization, and V_p is payload integrity. A release decision is

$V(C, t) = V_e \wedge V_a \wedge V_p$. (12)

The separation matters operationally. A consent revocation makes V_y , false even though the content and vector are unchanged. A corrupted vector makes V_p false even though the source and authorization remain valid. A deleted source makes both freshness and authorization fail, while deep regeneration may be impossible.

A production API should return a reason code rather than only a Boolean: `SOURCE_MISMATCH`, `SOURCE_MISSING`, `MODEL_MISMATCH`, `GOVERNANCE_DENY`, `PAYLOAD_MISMATCH`, or `VALID`. Reason codes permit deterministic actions, metrics by failure class, and audit evidence that explains why retrieval was blocked.

C. Temporal Semantics

The predicate must be evaluated against a coherent time point. If a source update commits at t_1 , a governance catalog refresh arrives at t_2 , and the vector is regenerated at t_3 , a naive evaluator can observe combinations that never existed as a

committed business state. The contract registry should therefore carry monotonically ordered source and governance revisions. An event is applied only if its revision is newer than the registry revision; older events are recorded as duplicates or late arrivals.

A practical state machine uses `VALID`, `BLOCKED`, `REFRESH_PENDING`, `DELETE_PENDING`, `QUARANTINED`, and `RETIRED`. A source edit moves a contract from `VALID` to `BLOCKED` or `REFRESH_PENDING`; successful regeneration installs a new vector and signature atomically and returns it to `VALID`. Consent revocation moves directly to `BLOCKED`, independent of deletion-job completion. A deleted source moves to `DELETE_PENDING`; no refresh path is permitted unless governance explicitly authorizes restoration.

APPENDIX B REGISTRY SCHEMA AND ENFORCEMENT ALGORITHMS

A. Minimum Durable Registry

The registry is the authoritative binding, not another optional metadata table. A minimal implementation needs stable identifiers, immutable evidence captured at embedding time, current policy state, lifecycle status, and action history. Table VII presents a storage-neutral schema. In a lakehouse, the fields can reside in an Iceberg/Delta table; in a vector database, a subset may be duplicated as filterable metadata, but the durable audit record should remain transactional.

Two fields are deliberately stronger than the benchmark's lightweight tuple: `artifact_digest` and `embedding_checksum`. The experiment omits them from the primary evaluator to expose the observation gap. A mature implementation should include both. Their cost is still materially below universal re-embedding and converts two deep-only scenarios into deterministic integrity checks.

B. Contract Evaluation

Algorithm 1 expresses the decision procedure. The order is intentional. Authorization and source existence are evaluated before expensive work. A governance denial must block retrieval even when source hashing or model services are unavailable. A source deletion must not trigger a re-embedding attempt. Only a contract that passes deterministic evidence is eligible for a deep check selected by policy. The transaction should prevent a time-of-check/time-of-use gap. One practical pattern is to store a compact contract token with the vector record: `contract_id`, `source_revision`, `governance_revision`, and `status`. The retrieval gateway admits a result only when the token matches a currently valid registry entry. Caching is allowed, but revocation events must invalidate the

authorization cache before or at the same time that the registry transitions to BLOCKED.

C. Atomic Refresh and Deletion

Regeneration is a versioned replacement, not an in-place overwrite with no history. A safe refresh sequence is: (1) compute the new canonical source and signature; (2) resolve the approved model artifact and digest; (3) create the new vector under a new physical key; (4) write the new contract version; (5) atomically switch the logical pointer; and (6) tombstone the prior vector. A failed step leaves the old vector blocked, not silently reactivated.

Deletion requires a different sequence. The read-path block is synchronous because it enforces authorization. Physical deletion from the vector store, caches, replicas, and backups can be asynchronous according to policy, but each stage must emit evidence. A deletion record should retain non-content audit fields such as contract identifier, event revision, reason, action status, and timestamps, while avoiding retention of the prohibited source or vector payload. Idempotency keys should be derived from `contract_id + governance_revision + action`. This prevents duplicate CDC or policy events from creating multiple vectors or conflicting delete jobs. A reconciliation job should compare active registry entries, vector-store keys, and index membership. Contracts with a valid registry state but no vector, or a vector with no active registry state, are integrity failures even when every individual system reports healthy.

Listing 1. Event-driven cross-modal contract evaluation.

```
function evaluate(contract_id, event):
  c <- registry.load_for_update(contract_id)
  if event.revision <=
    c.last_revision[event.domain]:
    return
  record("DUPLICATE_OR_LATE_EVENT"
  )
  current_g <-
  governance.resolve(c.source_key)
  if not authorized(current_g):
    registry.transition(c, BLOCKED,
    GOVERNANCE_DENY)
    enqueue(enforce_block_and_delete,
    c.vector_key)
  return INVALID
  current_r <- source.resolve(c.source_key)
  if current_r is MISSING:
    registry.transition(c,
    DELETE_PENDING, SOURCE_MISSING)
    enqueue(enforce_tombstone,
    c.vector_key)
```

```
return INVALID
current_sig <-
hash(canonicalize(current_r,
c.canonicalizer_version))
if current_sig != c.content_signature:
  registry.transition(c,
  REFRESH_PENDING,
  SOURCE_MISMATCH)
  enqueue(regenerate_embedding,
  c.contract_id)
return INVALID

active <-
model_registry.resolve(c.model_id)
if active.version != c.model_version:
  registry.transition(c,
  REFRESH_PENDING,
  MODEL_MISMATCH)
  enqueue(regenerate_embedding,
  c.contract_id)
return INVALID
if c.embedding_checksum exists and
checksum(vector.read(c.vector_key)) !=
c.embedding_checksum:
  registry.transition(c, QUARANTINED,
  PAYLOAD_MISMATCH)
return INVALID
if deep_policy.requires_verification(c,
event):
  if cosine(vector.read(c.vector_key),
  embed(active, current_r)) < c.threshold:
    registry.transition(c,
    QUARANTINED,
    PAYLOAD_MISMATCH)
  return INVALID
  registry.record_verification(c, current_sig,
  active.version, current_g.revision)
return VALID
```

APPENDIX C DETAILED EXPERIMENTAL EVIDENCE

A. Task-Level Allocation and Detector Overlap

The balanced allocation makes modality comparisons straightforward but also means aggregate recall is determined by scenario counts. Table VIII reports exact task-level outcomes. The signature detector's 72 true positives comprise 24 edits, 24 redactions, and 24 deletions. Governance also produces 72 true positives, but its coverage is redactions, deletions, and consent revocations. The two sets overlap on 48 contracts. Model-version checking adds 24 non-overlapping detections. Their union is therefore $72 + 72 + 24 - 48 = 120$, not 168.

The perfect balance across modalities produces identical detection counts for tabular, document,

and image data: each modality contributes 48 failures and 12 controls; lightweight evaluation detects 40 failures and deep verification detects 32; their union detects 48. This equality is a property of the experimental design, not evidence that modalities have equal operational risk. The latency results already show that images cost more to hash and regenerate. Production corpora also differ in object size, update frequency, encoder nondeterminism, and governance sensitivity.

B. Confidence Intervals and Paired Comparison

Point estimates on 144 failures and 36 controls should not be presented as population certainties. The 95% Wilson interval for lightweight recall (120/144) is 76.4-88.5%. The interval for deep-only recall (96/144) is 58.6-73.8%. The combined estimate is 100%, but its lower Wilson bound is 97.4%; finite testing cannot establish perfect future recall. Likewise, zero false positives among 36 controls yields a specificity interval of 90.4-100%, not proof of a zero false-alarm rate.

The lightweight and deep detectors are paired on the same failures. They agree on 72 detected contracts. Lightweight-only detection occurs on 48 contracts (deletion and consent), whereas deep-only detection occurs on 24 (payload corruption and unversioned artifact change). An exact two-sided McNemar test over the 72 discordant pairs gives $p = 0.0063$. Within this scenario mixture, the two methods are not interchangeable: lightweight evaluation covers significantly more failures because governance and source absence cannot be recovered by re-embedding. The comparison is not a claim that lightweight checks are intrinsically more accurate across arbitrary distributions; changing the frequency of payload failures would change the aggregate ranking.

C. Threshold Sensitivity

Table [IX expands the threshold result. “All-defect recall” counts every seeded failure, including governance states that no cosine comparison can detect. “Residual recall” is restricted to the 24 deep-only payload scenarios. At 0.999 and 0.9999, all residual failures are detected. Lower thresholds progressively accept changed vectors as sufficiently similar.

All controls have similarity 1.0 because the executable pipelines are deterministic and evaluated on the same software/hardware environment. This removes an important production noise source. Transformer inference, mixed precision, library upgrades, and different accelerators can produce small numerical differences. A production threshold must be calibrated from repeated clean regenerations for each model artifact and modality. Where deterministic equality is required, a signed

embedding checksum is preferable to a cosine heuristic.

D. Failure Walkthroughs

Source edit: A document’s operational text is changed while the vector key and stored embedding remain untouched. The current hash differs from S, so the lightweight evaluator blocks the contract without invoking the model. The deep verifier also observes a difference, but its high median cosine (0.99697) shows why a loose semantic threshold could miss small edits. The signature test answers a stronger question: whether the contracted content is identical under the declared canonicalization.

Redaction: A sensitive span is removed and the governance snapshot marks redaction. Both content and governance evidence change. The old vector must be blocked immediately because it may retain information correlated with the removed span. Regeneration is permitted only from the redacted source and must create a new contract version. Retaining the old vector

Model upgrade: The approved version changes from 1.0 to 2.0. Version mismatch is sufficient to invalidate the old contract even if some individual vectors remain numerically close. Mixed embedding spaces can alter nearest-neighbor relationships, thresholds, and downstream ranking. The correct unit of upgrade is a compatibility domain: a new version may coexist only if the retrieval layer explicitly isolates or translates spaces.

Consent revocation: Source bytes, signature, and vector remain exactly unchanged; cosine similarity is 1.0. Only governance detects the failure. This is the clearest demonstration that embedding validity is not reducible to data freshness. A technically accurate representation can still be unauthorized.

Vector corruption: Noise changes the stored vector while the source and metadata remain fixed. Signature, version, and governance checks all pass. Deep regeneration exposes the mismatch. A production vector checksum would detect accidental corruption more directly, while deep checks remain valuable for semantic or pipeline-level discrepancies.

Unversioned artifact replacement: The executable feature pipeline changes but retains the declared version. This is a release-governance failure. Deep verification detects it because the current artifact produces a different vector. The durable preventive control is an immutable artifact digest covering weights, tokenizer/extractor, preprocessing code, normalization, and dimensionality-reduction parameters.

APPENDIX D DEPLOYMENT PATTERNS AND OPERATIONAL CONTROLS

A. Lakehouse-Centric Pattern

In a lakehouse-centric deployment, the contract registry is an ACID table partitioned by status or source domain. Table/object change events append evaluation requests. A stream processor resolves current source and governance state, writes a decision record, and emits enforcement commands keyed by vector identity. Snapshot semantics are valuable because an investigator can reconstruct which source and policy revisions were visible when a vector was admitted.

The vector store should not be treated as the system of record for governance. Filter metadata is useful for low-latency query gating, but it can lag, be partially updated, or be absent from an approximate index. The authoritative status belongs in a durable registry, while the retrieval gateway combines registry/cache authorization with vector-store filtering. Periodic reconciliation compares logical active contracts with physical vector keys and index counts.

B. Vector-Store-Centric Pattern

Some platforms colocate object references and vectors. The same contract semantics still apply. Co-location reduces lookup hops but does not automatically create source signatures, model-artifact identity, or governance transitions. The vector record should carry the contract token and payload checksum; source/object changes should be translated into contract events. Where the vector store supports transactional metadata updates but not transactional object deletion, retrieval must honor the blocked status before asynchronous cleanup.

C. Read-Path Gate

The strongest enforcement point is the read path because background repair can fail or be delayed. For each candidate vector, the gateway checks an authorization cache populated from the registry. A blocked, pending-delete, quarantined, or unknown contract is excluded before content is returned to search, RAG, or an ML feature consumer. Fail-closed behavior is appropriate for high-risk PII or revoked consent; lower-risk analytical systems may define a short grace window for source refresh, but that exception must be explicit and measurable.

The gate should expose reason-coded metrics: blocked retrievals, stale-signature detections, version mismatches, governance denials, orphan vectors, deep-check failures, regeneration backlog, and deletion age. Infrastructure health such as index availability and query latency is necessary but orthogonal. A healthy index can serve invalid vectors efficiently.

D. Risk-Tiered Verification Policy

Universal deep verification is wasteful, while never inspecting payloads leaves a known gap. A practical policy combines triggers and sampling: verify every artifact release and model alias change;

- e verify all high-risk PII, regulated decisions, or deletionsensitive collections;

- e verify after storage repair, replication incident, or checksum discrepancy;

- e sample unchanged contracts by age, update frequency, and business impact;

- e increase sampling when drift or corruption rates exceed control limits;

- « re-verify before major retrieval-index rebuilds or migrations.

The sampling probability should be recorded so estimated defect prevalence can be corrected for selection bias. Deep verification output should not overwrite the contracted vector until the policy decision succeeds; otherwise the checker can accidentally repair evidence before the discrepancy is recorded.

E. Governance Action Matrix

F. Service-Level Objectives

A governance-sensitive deployment should define separate SLOs for detection, blocking, and physical remediation. For example, source edits may permit a refresh window, while consent revocation and deletion require near-immediate retrieval denial. The paper does not prescribe universal durations because legal basis and system architecture differ. It does prescribe measurable clocks: event occurrence, event ingestion, registry transition, read-path enforcement, vector deletion, cache invalidation, and verification completion.

Backlog metrics are essential. A system can report 100% detection while accumulating stale vectors that remain blocked but never regenerated, or deletion jobs that never complete. Operational completeness is the fraction of invalid contracts that reach the policy-prescribed terminal state within the SLO, not merely the fraction that raise an alert.

APPENDIX E MODEL SUBSTITUTION AND EXTERNAL REPLICATION PROTOCOL

A. What Changes When the Encoder Changes

The contract framework is independent of the local feature pipelines, but the deep-verification behavior is not. Replacing TF-IDF/SVD or HOG/PCA with a transformer changes inference cost, determinism, preprocessing, vector dimension, and the distribution of clean-run similarity. It does not change the logical meaning of source freshness, declared model compatibility, or governance authority. A source signature mismatch remains a mismatch without invoking the encoder; a revoked consent state remains a denial even when a state-of-the-art model would reproduce the vector exactly.

A credible external replication should therefore report two layers separately. The first layer is contract-mechanism performance: whether source, version, and governance evidence correctly identify invalid bindings. The second is encoder-dependent deep performance: whether regeneration and comparison detect payload or undeclared-pipeline changes at an acceptable false-positive rate. Combining the layers into one undifferentiated “accuracy” obscures which result is portable.

For a transformer text encoder, the artifact identity must cover model repository and immutable revision, tokenizer files, maximum sequence length, chunking and overlap, text extraction and normalization, pooling method, output normalization, quantization, and serving runtime. For a vision encoder, it must additionally cover decode library, color space, resize/crop policy, interpolation, channel normalization, image orientation handling, and precision. For a joint multimodal encoder, the text and image towers, projection heads, temperature or scaling parameters, and modality-specific preprocessing all belong to one compatibility domain. A version label that names only the base architecture is insufficient.

B. Proposed Replication Design

A follow-on study should use at least two independently implemented encoders per modality and retain the same 180-contract scenario allocation so detector counts remain comparable. Each clean object should be embedded repeatedly under: (1) the same process and hardware; (2) a restarted process; (3) CPU and GPU where supported; (4) full and reduced precision; and (5) at least two batch sizes. These repetitions estimate the null distribution of cosine distance when no semantic or governance change occurred. Thresholds must be chosen from that distribution before evaluating seeded failures; selecting a threshold after observing failures would bias recall upward.

The model-upgrade scenario should distinguish three cases.

A declared incompatible upgrade changes the version and requires regeneration. A declared compatible patch changes the artifact digest but is permitted only if a compatibility test passes. An undeclared replacement changes the artifact behind the same identity and is a governance failure regardless of average retrieval quality. This three-way design is more informative than treating every model update as drift.

The replication should report contract recall and specificity, payload-check recall, clean-run false-positive rate, inference latency, registry-evaluation latency, storage overhead per contract, and

enforcement completion time. Results should be stratified by modality, object size, chunk count, and governance class. For managed APIs, the study must record request parameters and response model identifiers because the executable artifact may not be available for hashing.

The expected result is not necessarily that deep verification reaches 100%. Semantically small edits can leave vectors nearly unchanged, and benign numerical variation can lower similarity. The lightweight contract offers exact evidence for identity and authority; deep verification offers probabilistic evidence about

the payload. A stronger production design combines both with embedding checksums and immutable artifact digests, reducing the number of questions delegated to a similarity threshold.

APPENDIX F REPRODUCIBILITY PROTOCOL AND ARTIFACT INVENTORY

A. Executable Environment

The released benchmark was executed with Python 3.13.5 on Linux 6.12.13 (x86-64), an Intel Xeon Platinum 8573C allocation with five logical CPUs, and approximately 5.9 GiB of RAM. The principal packages were NumPy 2.3.5, pandas 2.2.3, scikit-learn 1.8.0, Pillow 12.2.0, and scikit-image 0.26.0. The benchmark uses a fixed random seed and local deterministic models. No external embedding API, proprietary model, or production corpus is required. The code generates source artifacts, fits model versions, creates the embedding-time registry, seeds current-state scenarios, evaluates all detector components, measures repeated execution time, and exports figures and tabular results. Raw generated objects are included so readers can inspect the exact canonical inputs. Embedding vectors are released as numeric rows linked by contract identifier.

B. Reproduction Steps

A clean reproduction follows these steps:

- 1) create an isolated Python environment with the listed package versions;
- 2) run `code/run_benchmark.py` from the package root;
- 3) verify that 180 contracts, 144 defects, and 36 controls are emitted;
- 4) compare `scenario_summary.csv`, `latency_summary.csv` with the reported tables;
- 5) run `code/make_figures.py` to regenerate all six figures;
- 6) inspect `manifest.json` for environment, seed, model, and artifact metadata.

Timing values will vary by hardware and package implementation. Exact detector counts should remain stable under thereleased versions because scenario generation, canonicalization, and model fitting are seeded and deterministic. If counts

differ, the correct response is to investigate the artifact or dependency change, not to update the paper's numbers silently.

C. Reproducibility Boundaries

The release supports computational reproduction of the reported controlled benchmark, not external validation of enterprise compliance. It does not include proprietary lakehouse connectors, commercial vector-store behavior, production IAM systems, or legal-policy interpretation. The generated corpus is intentionally simple enough to inspect. The empirical claim is limited to the seeded failures and executable pipelines in the package.

The local models were selected to isolate lifecycle behavior. A follow-on replication should substitute transformer text encoders and vision-language models, repeat clean inference across CPU/GPU and precision modes, and calibrate model-specific thresholds. It should also introduce object transcoding, chunk-boundary changes, extraction-library upgrades, partial batch failures, out-of-order governance events, and index replication lag. Those tests would measure how much of the present deterministic behavior survives under production complexity.

D. Decision Checklist

Before an organization calls an embedding “governed,” it should be able to answer the following with evidence:

e Which exact source revision, canonicalizer, model artifact, and governance revision produced this vector?

¢ Can a source edit, redaction, deletion, or consent change identify every affected vector key?

¢ Does retrieval fail closed while regeneration or deletion is pending?

e Are model weights, tokenizer/extractor, preprocessing, normalization, and projection covered by one immutable artifact digest?

e Can the system detect orphan vectors and missing index entries by reconciliation?

e Which failure classes are checked deterministically, which are sampled, and which remain accepted residual risk?

e Are detection, block, regeneration, deletion, and verification timestamps retained for audit?

A negative answer identifies a concrete control gap. The value of the tuple (R, E, S, V, G) is that it converts a vague expectation—“the vector should stay in sync”—into named evidence, executable predicates, and enforcement responsibilities.

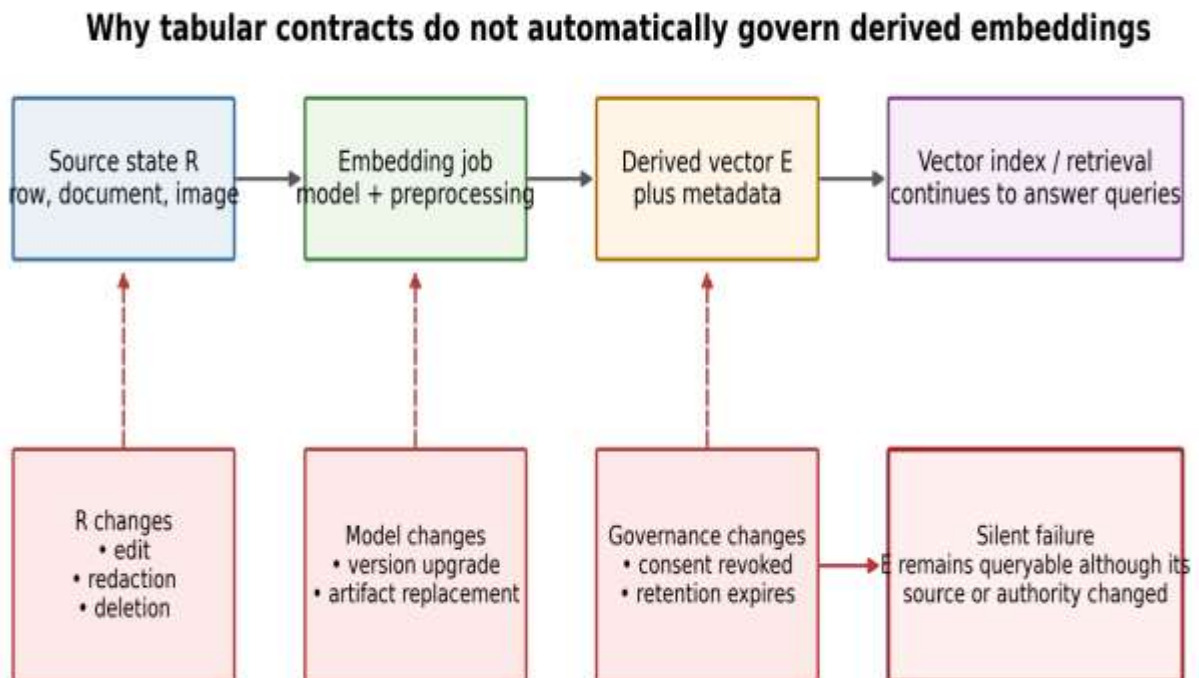
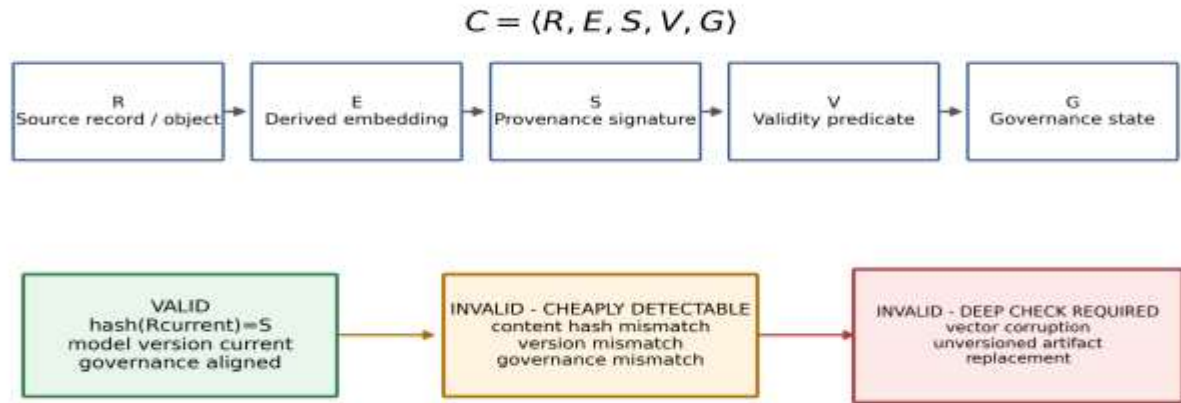


Figure. 1. Silent cross-modal drift. Conventional health checks can remain green while the source, model, or governance authority changes and the old embedding remains queryable.



The lightweight predicate tests provenance and authority; selective re-embedding tests the vector payload itself.

Figure 2. Formal contract and validity states. The lightweight predicate evaluates source provenance, model compatibility, and governance authority. Re-embedding is reserved for failures invisible to those fields.

Table 1 Cross-modal contract components and enforcement semantics

Component	Stored evidence	Contract question	Example break	Required response
R	Source key, URI, snapshot/object identity	What current record or object is represented?	Object deleted or row superseded	Resolve current state; block use if missing
E	Vector, dimension, index key	Which derived representation is being served?	Vector payload corrupted	Quarantine and deep-verify or regenerate
S	SHA-256 of canonical source at embedding time	Has the represented content changed?	Edit or redaction without regeneration	Mark stale; regenerate or delete
V	Executable equality and policy predicates	Is the binding currently valid?	Version tag differs from approved model	Route to refresh workflow
G	Consent, PII class, retention, deletion, redaction	Is retention and use still authorized?	Consent revoked but vector remains queryable	Tombstone/delete and record audit evidence

Proposed Cross-Modal Contract Enforcement Architecture

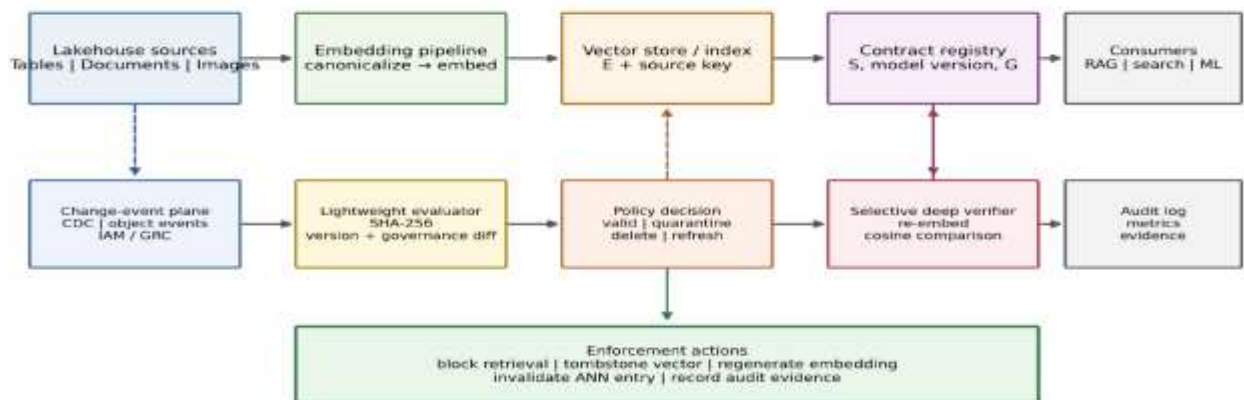


Figure 3. Proposed architecture. The registry binds source, vector, signature, model version, and governance state; the event plane drives cheap evaluation, while selective re-embedding is an escalation rather than a universal tax.

Table 2. benchmark composition and seeded failure mechanisms

Scenario	Per modality	Total	Seeded state transition	Expected detector
Unchanged control	12	36	No source, model, governance, or vector change	None
Source edit	8	24	Content changed; embedding retained	Signature; deep
Source redaction	8	24	PII-bearing content altered; old vector retained	Signature/governance; deep
Source deletion	8	24	Source unavailable and deletion flag set; vector retained	Signature/governance
Model upgrade	8	24	Approved version changed from 1.0 to 2.0; old vector retained	Model tag; deep
Consent revocation	8	24	Current consent false; vector governance snapshot remains true	Governance
Vector corruption	4	12	Noise injected into stored vector; metadata unchanged	Deep only
Unversioned model change	4	12	Active embedding artifact replaced while version remains 1.0	Deep only

Table 3 experimental embedding model inventory

Modalities	Declared model	Version	Executable representation pipeline	Role
Tabular, document	local-lsa	1.0	TF-IDF word 1–2 grams (512) → SVD-48 → L2 normalize	Baseline stored embedding
Tabular, document	local-lsa	2.0	TF-IDF character 3–5 grams (768) → SVD-48 → L2 normalize	Declared model upgrade
Tabular, document	local-lsa	1.0 (artifact 1b)	Binary word 1–3 grams (640) → SVD-48 → L2 normalize	Undeclared artifact change
Image	local-hog-pca	1.0	HOG-9/16px + RGB histograms → PCA-32 → L2 normalize	Baseline stored embedding
Image	local-hog-pca	2.0	HOG-12/8px + higher-resolution RGB histograms → PCA-32	Declared model upgrade
Image	local-hog-pca	1.0 (artifact 1b)	HOG-6/8px/3×3 blocks + grayscale histogram → PCA-32	Undeclared artifact change

Pre-production Benchmark Workflow

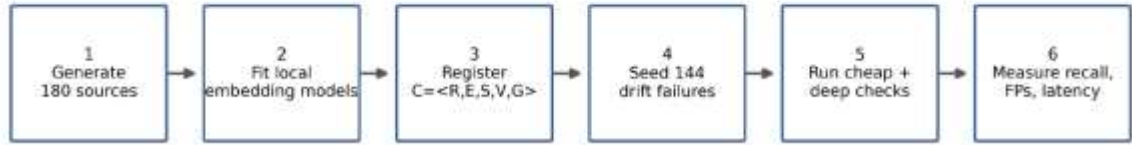


Figure 4. Experimental workflow. The benchmark records task-level ground truth and times actual contract evaluation and re-embedding operations.

Table 4 aggregate detection results on 180 contracts

Detector	TP	FN	TN	FP	Recall (%)	Specificity (%)	Precision (%)	Accuracy (%)
Signature check	72	72	36	0	50.0	100.0	100.0	60.0
Model-version check	24	120	36	0	16.7	100.0	100.0	33.3
Governance check	72	72	36	0	50.0	100.0	100.0	60.0
Lightweight contract union	120	24	36	0	83.3	100.0	100.0	86.7
Deep re-embedding only	96	48	36	0	66.7	100.0	100.0	73.3
Combined policy	144	0	36	0	100.0	100.0	100.0	100.0

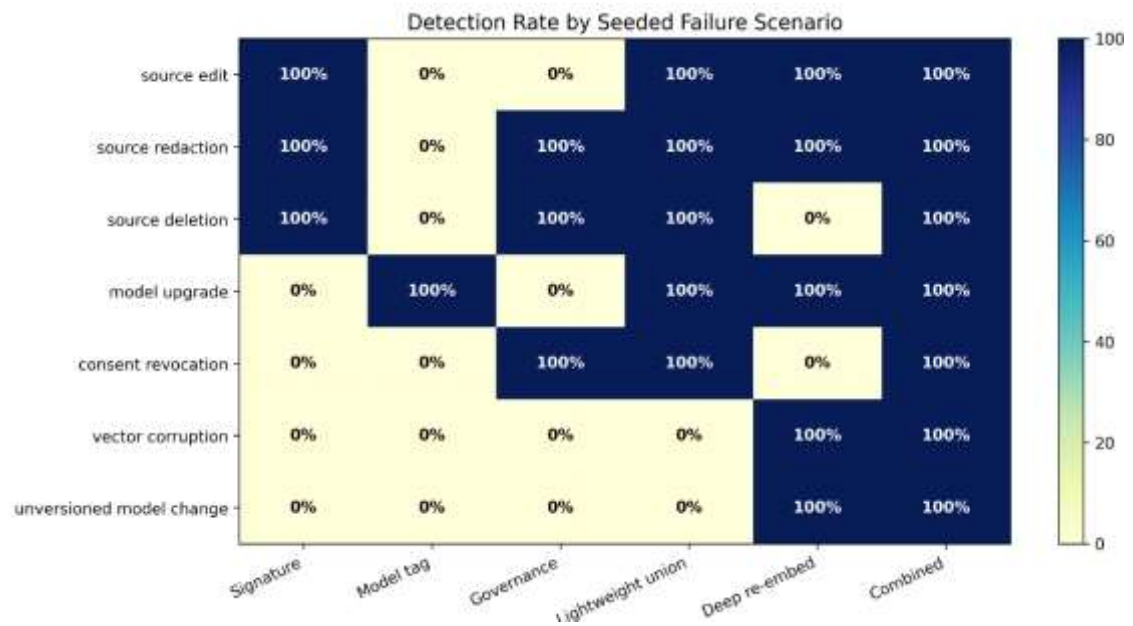


Figure 5. Scenario-level detection. No single detector covers source, model, governance, and payload failures.

TABLE V
OBSERVED PER-CONTRACT LATENCY

Modality	Lightweight median	Deep median	Ratio
Tabular	6.83 μ s	0.424 ms	62.0×
Document	6.48 μ s	0.432 ms	66.6×
Image	27.81 μ s	2.839 ms	102.1×

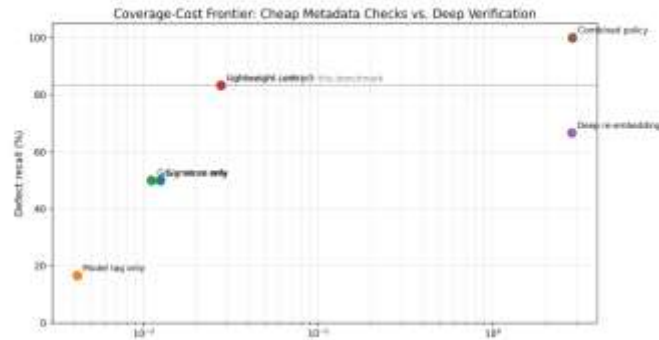


Figure 6. Coverage-cost frontier. Metadata checks are inexpensive but have a defined observation ceiling; deep verification closes payload-level gaps at higher cost

Table 6 detector observability and residual-risk matrix

Failure class	Signature	Declared version	Governance	Deep re-embed	Evidence needed for cheap deterministic detection
Source edit	Yes	No	No	Yes	Canonical source signature
Source redaction	Yes	No	Usually	Yes	Signature plus redaction/PII state
Source deletion	Yes/existence	No	Yes	Not executable	Existence/tombstone and deletion authority
Declared model upgrade	No	Yes	No	Yes	Immutable approved-version identifier
Consent revocation	No	No	Yes	No	Current authorization state on the read path
Vector corruption	No	No	No	Yes	Vector checksum, storage checksum, or deep verification
Unversioned artifact replacement	No	No	No	Yes	Signed artifact digest and preprocessing fingerprint
Index entry omission/duplication	No	No	No	Not necessarily	Index reconciliation keyed by contract/vector identity

Table 7 minimum contract-registry schema

Field	Type	Set by	Semantics	Integrity rule
-------	------	--------	-----------	----------------

contract_id	UUID/text	Contract service	Stable identity across refreshes	Unique, immutable
source_key	Text	Source adapter	Business/object identifier	Resolves within declared source snapshot
source_revision	Text/int	CDC/object event	Monotonic source state	Reject older revisions
modality	Enum	Source adapter	Tabular, document, image, audio, other	Closed controlled vocabulary
vector_key	Text	Embedding service	Vector-store/index identity	Unique active key per contract version
content_signature	Bytes/text	Canonicalizer	Hash of canonical source at embedding time	Algorithm and canonicalizer version recorded
model_id	Text	Model registry	Logical representation family	Must reference approved model
model_version	Text	Model registry	Declared compatible version	Immutable for a generated vector
artifact_digest	Bytes/text	Build/registry	Digest of weights, tokenizer, extractor, and configuration	Signed or registry-attested
embedding_checksum	Bytes/text	Embedding service	Digest of serialized vector payload	Recomputed at write/read verification
governance_revision	Text/int	Policy service	Ordering of consent/retention/PII state	Reject stale governance events
governance_snapshot	JSON/struct	Policy service	Authority captured at generation time	Versioned policy schema
status	Enum	Contract service	Valid, blocked, pending, quarantined, retired	Transitions constrained by state machine
created_at / verified_at	Timestamp	Contract service	Evidence age and operational SLO	UTC, monotonic within contract history
reason_code	Enum	Evaluator	Last decision explanation	Required whenever status is not valid

Table 8 exact scenario-level detection counts

Scenario	Defects	Signature	Model	Governance	Lightweight	Deep	Combined	Median cosine
Source edit	24	24	0	0	24	24	24	0.99697
Source redaction	24	24	0	24	24	24	24	0.96436
Source deletion	24	24	0	24	24	0	24	N/A
Model upgrade	24	0	24	0	24	24	24	0.74488
Consent revocation	24	0	0	24	24	0	24	1.00000
Vector corruption	12	0	0	0	0	12	12	0.91943
Unversioned	12	0	0	0	0	12	12	0.96389

model change								
Total	144	72	24	72	120	96	144	—

Table 9 deep-verification threshold sensitivity

τ	All TP	All recall	Residual TP	Residual recall
0.9999	96	66.7%	24	100.0%
0.9990	96	66.7%	24	100.0%
0.9950	72	50.0%	21	87.5%
0.9900	68	47.2%	20	83.3%
0.9800	68	47.2%	20	83.3%
0.9500	50	34.7%	11	45.8%
0.9000	32	22.2%	5	20.8%

Table 10 recommended enforcement actions by contract failure

Failure	Immediate read action	Physical action	Required audit evidence	Safe return to service
Source edit	Block or mark stale	Generate new vector; tombstone old	Old/new source revisions and signatures	New vector, signature, and pointer committed
Redaction	Block immediately	Delete old vector; embed redacted source if authorized	Redaction revision, affected contract keys, deletion result	Only redacted-source vector active
Source deletion	Block immediately	Tombstone/delete vector and cache entries	Deletion authority, key inventory, completion timestamps	Only after explicit lawful restoration
Declared model upgrade	Isolate old compatibility domain or block	Batch/event-driven regeneration	Old/new artifact digests and coverage progress	Contract points to approved compatible version
Consent revocation	Deny immediately	Delete/tombstone according to policy	Consent revision, block timestamp, deletion status	New consent/lawful basis and new contract version
Payload corruption	Quarantine	Restore from verified regeneration	Checksum/deep evidence and storage incident link	Payload checksum and deep comparison pass
Unversioned artifact change	Freeze release and quarantine affected range	Restore approved artifact or regenerate all affected vectors	Artifact digest, deployment history, affected contract set	Immutable version/digest registered and verified

Table 11 recommended external replication matrix for production-class encoders

Encoder class	Example experimental role	Artifact evidence that must be frozen	Required clean-run calibration	Failure injections to repeat
Sentence transformer	Document and serialized-record embeddings	Weights revision, tokenizer, chunker, pooling, normalization, precision, runtime	Repeated same-input inference across CPU/GPU and batch sizes	Edit, redaction, model revision, tokenizer change, vector corruption, alias replacement
General language model encoder	Long or domain text representation	Checkpoint, tokenizer vocabulary, truncation, prompt/template, hidden-layer selection	Similarity and checksum stability by sequence length	Extraction change, chunk-boundary shift, quantization change, undeclared checkpoint swap

Vision encoder	Image/object embeddings	Weights, decoder, EXIF handling, resize/crop, interpolation, color normalization	Same-file and visually equivalent-file distributions	Pixel edit, metadata-only rewrite, transcoding, crop change, corruption, model upgrade
Vision-language encoder	Shared text-image retrieval space	Both towers, projection heads, tokenizer, image processor, scaling/temperature	Within-modality and cross-modality stability	One-tower upgrade, preprocessing mismatch, joint-space mixing, artifact alias change
Managed embedding API	Enterprise service replication	Provider model identifier, dated API version, dimensions, region, request parameters	Repeated requests over time and regions; provider-change monitoring	Silent provider upgrade, dimension change, rate-limit fallback, partial batch retry

Table 12. supplementary artifact inventory

Artifact	Granularity	Purpose
source_catalog.csv and raw_objects/	180 sources	Canonical source metadata and generated text/PNG objects
contract_registry.csv	180 contracts	Embedding-time source signature, model, vector key, and governance snapshot
embeddings.csv	180 vectors	Stored vector payloads linked to contract identity
benchmark_results.csv	180 outcomes	Ground truth, component detections, union, deep result, similarity, and timing
scenario_summary.csv	8 scenarios	Exact detector counts and median similarity by failure class
aggregate_metrics.csv	6 detectors	TP, FN, TN, FP, recall, specificity, precision, and accuracy
latency_summary.csv	3 modalities	Median and 95th-percentile lightweight/deep latency and ratio
threshold_sensitivity.csv	14 rows	Deep-check outcomes across seven thresholds and two evaluation scopes
governance_events.csv	Governance scenarios	Current and stored consent, redaction, deletion, and action state
run_benchmark.py	Executable code	Corpus generation, model fitting, fault seeding, evaluation, export
make_figures.py	Executable code	Reproduction of six publication figures
manifest.json	Package level	Seed, environment, model inventory, counts, and generation timestamp

Author Statements:

- **Ethical approval:** The conducted research is not related to either human or animal use.
- **Conflict of interest:** The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper

- **Acknowledgement:** The authors declare that they have nobody or no-company to acknowledge.
- **Author contributions:** The authors declare that they have equal right on this paper.
- **Funding information:** The authors declare that there is no funding to be acknowledged.
- **Data availability statement:** The data that support the findings of this study are available

on request from the corresponding author. The data are not publicly available due to privacy or ethical restrictions.

- **Use of AI Tools:** The author(s) declare that no generative AI or AI-assisted technologies were used in the writing process of this manuscript.

References

- [1] R. Y. Wang and D. M. Strong, "Beyond accuracy: What data quality means to data consumers," *J. Manage. Inf. Syst.*, vol. 12, no. 4, pp. 5–33, 1996, doi: 10.1080/07421222.1996.11518099.
- [2] C. Batini, C. Cappiello, C. Francalanci, and A. Maurino, "Methodologies for data quality assessment and improvement," *ACM Comput. Surv.*, vol. 41, no. 3, Art. no. 16, 2009, doi: 10.1145/1541880.1541883.
- [3] Y. L. Simmhan, B. Plale, and D. Gannon, "A survey of data provenance in e-science," *SIGMOD Rec.*, vol. 34, no. 3, pp. 31–36, 2005, doi: 10.1145/1084805.1084812.
- [4] P. Buneman, S. Khanna, and W.-C. Tan, "Why and where: A characterization of data provenance," in *Database Theory—ICDT 2001*, pp. 316–330, doi: 10.1007/3-540-44503-X_20.
- [5] J. Cheney, L. Chiticariu, and W.-C. Tan, "Provenance in databases: Why, how, and where," *Found. Trends Databases*, vol. 1, no. 4, pp. 379–474, 2009, doi: 10.1561/19000000006.
- [6] L. Moreau et al., "The Open Provenance Model core specification (v1.1)," *Future Gener. Comput. Syst.*, vol. 27, no. 6, pp. 743–756, 2011, doi: 10.1016/j.future.2010.07.005.
- [7] A. M. Chirigati, R. Rampin, D. Shasha, and J. Freire, "ReproZip: Computational reproducibility with ease," in *Proc. ACM SIGMOD*, 2016, pp. 2085–2088, doi: 10.1145/2882903.2899401.
- [8] M. Boehm et al., "Automated provenance tracking in data science scripts," in *Proc. 26th ACM SIGKDD*, 2020, pp. 2510–2518, doi: 10.1145/3394486.3403205.
- [9] M. Armbrust et al., "Delta Lake: High-performance ACID table storage over cloud object stores," *Proc. VLDB Endow.*, vol. 13, no. 12, pp. 3411–3424, 2020, doi: 10.14778/3415478.3415560.
- [10] S. Ait Errami, H. Hajji, K. Ait El Kadi, and H. Badir, "Spatial big data architecture: From data warehouses and data lakes to the LakeHouse," *J. Parallel Distrib. Comput.*, vol. 176, pp. 70–79, 2023, doi: 10.1016/j.jpdc.2023.02.007.
- [11] J. Wang et al., "Milvus: A purpose-built vector data management system," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 2021, pp. 2614–2627, doi: 10.1145/3448016.3457550.
- [12] M. Aumüller, E. Bernhardsson, and A. Faithfull, "ANN-Benchmarks: A benchmarking tool for approximate nearest neighbor algorithms," *Inf. Syst.*, vol. 87, Art. no. 101374, 2020, doi: 10.1016/j.is.2019.02.006.
- [13] Y. A. Malkov and D. A. Yashunin, "Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 42, no. 4, pp. 824–836, 2020, doi: 10.1109/TPAMI.2018.2889473.
- [14] J. Johnson, M. Douze, and H. Jégou, "Billion-scale similarity search with GPUs," *IEEE Trans. Big Data*, vol. 7, no. 3, pp. 535–547, 2021, doi: 10.1109/TBDATA.2019.2921572.
- [15] N. Reimers and I. Gurevych, "Sentence-BERT: Sentence embeddings using Siamese BERT-networks," in *Proc. EMNLP-IJCNLP*, 2019, pp. 3982–3992, doi: 10.18653/v1/D19-1410.
- [16] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proc. NAACL-HLT*, 2019, pp. 4171–4186, doi: 10.18653/v1/N19-1423.
- [17] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE CVPR*, 2016, pp. 770–778, doi: 10.1109/CVPR.2016.90.
- [18] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Proc. IEEE CVPR*, 2005, pp. 886–893, doi: 10.1109/CVPR.2005.177.
- [19] N. Halko, P.-G. Martinsson, and J. A. Tropp, "Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions," *SIAM Rev.*, vol. 53, no. 2, pp. 217–288, 2011, doi: 10.1137/090771806.
- [20] I. T. Jolliffe, *Principal Component Analysis*, 2nd ed. New York, NY, USA: Springer, 2002, doi: 10.1007/b98835.
- [21] M. Mitchell et al., "Model cards for model reporting," in *Proc. FAT**, 2019, pp. 220–229, doi: 10.1145/3287560.3287596.
- [22] T. Gebru et al., "Datasheets for datasets," *Commun. ACM*, vol. 64, no. 12, pp. 86–92, 2021, doi: 10.1145/3458723.
- [23] E. M. Bender and B. Friedman, "Data statements for natural language processing: Toward mitigating system bias and enabling better science," *Trans. Assoc. Comput. Linguist.*, vol. 6, pp. 587–604, 2018, doi: 10.1162/tacl_a_00041.
- [24] P. Domingos, "A few useful things to know about machine learning," *Commun. ACM*, vol. 55, no. 10, pp. 78–87, 2012, doi: 10.1145/2347736.2347755.
- [25] E. Breck, S. Cai, E. Nielsen, M. Salib, and D. Sculley, "The ML test score: A rubric for ML production readiness and technical debt reduction," in *Proc. IEEE Int. Conf. Big Data*, 2017, pp. 1123–1132, doi: 10.1109/BigData.2017.8258038.
- [26] N. Sambasivan et al., "Everyone wants to do the model work, not the data work: Data cascades in high-stakes AI," in *Proc. CHI*, 2021, doi: 10.1145/3411764.3445518.
- [27] S. Zhang, A. Bouadi, and K. Werder, "Establishing data provenance for responsible artificial intelligence systems," *ACM Trans. Manage. Inf. Syst.*, vol. 13, no. 2, Art. no. 22, 2022, doi: 10.1145/3503488.

- [28] M. Veale, R. Binns, and L. Edwards, "Algorithms that remember: Model inversion attacks and data protection law," *Phil. Trans. R. Soc. A*, vol. 376, Art. no. 20180083, 2018, doi: 10.1098/rsta.2018.0083.
- [29] Y. Cao and J. Yang, "Towards making systems forget with machine unlearning," in *Proc. IEEE Symp. Security Privacy*, 2015, pp. 463–480, doi: 10.1109/SP.2015.35.
- [30] C. Guo, T. Goldstein, A. Hannun, and L. van der Maaten, "Certified data removal from machine learning models," in *Proc. ICML*, 2020, pp. 3832–3842, doi: 10.48550/arXiv.1911.03030.
- [31] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership inference attacks against machine learning models," in *Proc. IEEE Symp. Security Privacy*, 2017, pp. 3–18, doi: 10.1109/SP.2017.41.
- [32] M. Fredrikson, S. Jha, and T. Ristenpart, "Model inversion attacks that exploit confidence information and basic countermeasures," in *Proc. ACM CCS*, 2015, pp. 1322–1333, doi: 10.1145/2810103.2813677.
- [33] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation," *ACM Comput. Surv.*, vol. 46, no. 4, Art. no. 44, 2014, doi: 10.1145/2523813.
- [34] E. Caveness et al., "TensorFlow Data Validation: Data analysis and validation in continuous ML pipelines," in *Proc. ACM SIGMOD*, 2020, pp. 2793–2796, doi: 10.1145/3318464.3384707.
- [35] S. Schelter, D. Lange, P. Schmidt, M. Celikel, F. Biessmann, and A. Grafberger, "Automating large-scale data quality verification," *Proc. VLDB Endow.*, vol. 11, no. 12, pp. 1781–1794, 2018, doi: 10.14778/3229863.3229867.
- [36] C. Wohlin et al., *Experimentation in Software Engineering*. Berlin, Germany: Springer, 2012, doi: 10.1007/978-3-642-29044-2.
- [37] R. Gollapudi, "Operational drift and risk-bounded decision-making in production database systems," *J. Int. Crisis Risk Commun. Res.*, vol. 6, no. S3, pp. 132–147, 2023, doi: 10.63278/jicrcr.vi.3762.