International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING
ISSN:2147-6799 www.ijisae.org Original Research Paper

Contrastive Failure Embeddings for Lakehouse Reliability Engineering

Mogana Kumaran Sivaraman

Submitted: 25/06/2023 **Revised:** 02/08/2023 **Accepted:** 12/08/2023

Abstract: This study proposes a contrastive embedding approach for reliability engineering in lakehouse environments. The objective is to learn a vector representation of failure episodes that captures operational similarity, enabling recurrence detection, nearest-neighbor retrieval, and similarity-based analysis of historical incidents. Failure records are constructed from normalized telemetry features reflecting execution behavior, resource usage, runtime outcomes, and contextual metadata. A neural encoder is trained using a supervised contrastive loss with temperature-controlled scaling, pulling same-class failure pairs together while pushing apart pairs from different classes to produce geometrically well-separated clusters suited to cosine-similarity retrieval. Because the encoder maps failures into a continuous space rather than assigning discrete class labels, new failure types can be accommodated without retraining: a novel failure occupies a previously unoccupied region of the space, preserving the open-set flexibility that production environments require. The method extends deterministic failure fingerprinting with a learned representation layer for cases where exact matching is insufficient, organizing failures by operational similarity rather than surface-level error attributes. Contrastive failure embeddings offer a practical mechanism for improving incident retrieval in lakehouse environments and a foundation for intelligent operational assistants.

Keywords: *contrastive learning, failure embeddings, lakehouse reliability, incident retrieval*

1. Introduction

Lakehouse platforms execute diverse analytical and ML workloads at scale, and the operational failures that arise are correspondingly varied. A single platform may host batch ETL, streaming ingestion, interactive queries, and model training, each with distinct resource profiles, failure modes, and remediation requirements. Identifying whether a newly observed failure is a recurrence of a known issue or a genuinely novel incident is a fundamental challenge for reliability engineering.

Two strategies dominate today. Deterministic approaches rely on exact matching of error codes, exception types, and stack trace hashes. These are effective when failures reproduce identically but fail when the same underlying problem manifests with different surface characteristics. A schema validation error, for example, may produce different messages depending on column types, pipeline stage, and framework version. Deterministic fingerprints then treat these as distinct failures even when the root cause is the same.

The brittleness has many faces. The brittleness of deterministic fingerprinting becomes apparent through several concrete failure modes. Framework version upgrades frequently alter exception hierarchies: an OOM failure producing `java.lang.OutOfMemoryError: Java heap space` under one Spark runtime may instead surface as `GC overhead limit exceeded` or a container-level `OOMKilled` signal under a different configuration, even though the underlying cause is identical. Environment-specific details contaminate otherwise identical signatures. Cluster identifiers, timestamps, file paths, and partition names appearing within stack traces cause hash mismatches between operationally indistinguishable failures. Transient infrastructure conditions alter which exception propagates to the

Sr Staff Software Engineer

top-level error report, producing nondeterministic fingerprints for a single root cause.

Open-set is a related concern. Deterministic methods also face an open-set problem. Production lakehouse environments are not static: new data sources are onboarded, pipeline logic evolves, cluster configurations change, and platform upgrades introduce previously unseen failure modes. Any approach requiring an exhaustive, pre-enumerated catalog cannot recognize novel failure types at first occurrence. In a classifier, a failure type absent from the training labels is either misclassified or rejected. The open-set nature of operational failures demands a representation that accommodates new classes without invalidating the existing model. Retrieval over a continuous embedding space naturally provides that property.Learned representations offer a complementary approach. Projecting failure records into a continuous vector space allows operational similarity to be measured as a distance or angle rather than exact attribute match. Failures that are operationally related but superficially different can be identified through proximity in the embedding space.

Here is the proposal. This study presents a contrastive embedding framework for lakehouse reliability engineering. The framework constructs failure records from normalized execution telemetry, trains a neural encoder using supervised contrastive learning, and uses the resulting embeddings for recurrence detection and incident retrieval. Supervised contrastive learning directly optimizes grouping same-class failures together while separating classes, producing an embedding geometry suited to similarity-based retrieval.

The contributions of this study are:

1. A method for constructing structured failure records from lakehouse execution telemetry suitable for contrastive learning.
2. A supervised contrastive training procedure for producing failure embeddings that capture operational similarity.
3. A discussion of how contrastive embeddings extend deterministic fingerprinting and support more flexible recurrence detection.
4. An analysis of design considerations, including normalization strategy, embedding dimensionality, and the implications of the learned representation for operational workflows.
5.

2. Background and Related Work

2.1 Contrastive Learning

The idea is simple. Contrastive learning produces representations by training a model to distinguish similar pairs from dissimilar pairs. Semantically related inputs are mapped to nearby points in a representation space while unrelated inputs are mapped to distant points. The loss functions operate over positive pairs (inputs that should be similar) and negative pairs (inputs that should be dissimilar) within each training batch.

Two key losses anchor the literature. SimCLR [1] introduced the Normalized Temperature-scaled Cross-Entropy (NT-Xent) loss in the self-supervised setting. For a batch of N samples, each input is augmented twice to produce two views. For anchor i with its augmented view j(i) as the positive:

$$L_{NT\text{-}Xent}(i) = -\log \left[\exp(z_i \cdot z_{j(i)} / \tau) / \Sigma_{a \neq i} \exp(z_i \cdot z_a / \tau) \right]$$

Each anchor has exactly one positive (the other augmented view), while all remaining 2(N-1) samples are negatives. This single-positive formulation fits self-supervised learning where no labels are available.

Labels change the picture. Supervised contrastive learning [2] extends the contrastive framework to the labeled setting. Rather than treating only augmented views of the same input as positives, the Supervised Contrastive (SupCon) loss treats all samples sharing the same class label as positives for a given anchor:

$$L_{SupCon}(i) = -(1/|P(i)|) \cdot \Sigma_{p \in P(i)} \log \left[\exp(z_i \cdot z_p / \tau) / \Sigma_{a \neq i} \exp(z_i \cdot z_a / \tau) \right]$$

P(i) is the set of indices in the batch sharing the anchor's class label, excluding i itself. The key difference from NT-Xent is averaging over multiple positives: SupCon aggregates log-probability terms across all same-class samples. This richer gradient signal leads to tighter within-class clustering and more uniform between-class separation. Both properties benefit nearest-neighbor retrieval. Graf et al. [3] showed that the supervised contrastive loss reaches its minimum when class representations collapse to the vertices of a regular simplex on the unit hypersphere. This geometric property implies a well-trained encoder

positions each failure class at a distinct, maximally-separated point, supporting both classification and nearest-neighbor retrieval.

2.2 Contrastive Learning for Tabular Data

The mainstream lies elsewhere. Most contrastive learning research has focused on image and text, where natural augmentations are well established. Tabular data presents structural challenges with no direct analog: individually meaningful features, heterogeneous scales and types, and no inherent feature ordering.Despite these challenges, several methods have demonstrated contrastive representation learning on structured data. SCARF [4] generates positive views through random feature corruption, replacing a subset of features with values drawn from their empirical marginal distributions. VIME [5] uses mask-and-predict pretraining where the model reconstructs masked features while being trained on a downstream task. SubTab [6] generates contrastive views through feature subsetting, dividing the feature set into overlapping subsets. Shwartz-Ziv and Armon [7] examined deep learning for tabular data, finding that gradient-boosted tree ensembles remain competitive on many benchmarks and identifying conditions under which learned representations provide genuine advantages.

This study applies supervised contrastive learning to operational telemetry. Operational telemetry is a tabular domain where class labels correspond to failure types. Positive pairs are defined by shared labels rather than by data transformations, though feature heterogeneity remains relevant.

2.3 Failure Detection and Operational Intelligence

AIOps has matured over a decade. The application of ML to IT operations, commonly termed AIOps, has evolved through several phases. Early tooling relied on rule-based systems: static thresholds, regex log-pattern matching, and manually curated runbooks. These are transparent and fast but limited to anticipated failure modes.

The next phase introduced statistical and classical ML methods. DeepLog [8] models normal log sequences with LSTMs and detects anomalies. Drain [9] extracts log templates via a fixed-depth tree parser at scale. LogRobust [10] addresses log instability across software versions. These approaches improved coverage over rule-based methods but generally operated as classification or anomaly-detection systems, answering 'what type of failure is this?' without measuring similarity between failures.

More recent work has addressed fault analysis and intelligent operations at scale. Zhou et al. [11] surveyed fault analysis and debugging in microservice systems, establishing benchmarks for how failures propagate through distributed architectures. Dang et al. [12] articulated the real-world challenges and research innovations in AIOps, outlining how data-driven methods augment traditional rule-based workflows. These studies highlight the growing importance of automated operational intelligence but generally focus on classification and detection rather than similarity-based retrieval over continuous embedding spaces.

Retrieval changes the framing entirely. The retrieval framing used here occupies a distinct position. The objective is not only to classify the failure but to identify its nearest historical analogs in a continuous similarity space. Retrieval naturally supports open-set scenarios: new failure types can be added to the registry without retraining. It also provides a ranked list of similar historical incidents rather than a single classification label, enabling operators to inspect multiple analogs and judge the relevance of each.

2.4 Lakehouse Platforms and Storage Layers

Lakehouses are the platform context. Lakehouse architectures [13] combine data-warehouse management with data-lake flexibility. A key enabler is ACID-compliant table storage over cloud object stores, as in Delta Lake [14], which provides transactional guarantees, schema enforcement, and time travel. The operational telemetry these platforms generate covers job execution metrics, cluster resource usage, and runtime error information. It provides the raw material for constructing failure representations.

2.5 Approximate Nearest Neighbor Search

Search has to be fast. The retrieval component relies on efficient similarity search over the embedding registry. Approximate nearest neighbor (ANN) algorithms make this feasible at scale. Johnson et al. [15] demonstrated billion-scale similarity search with GPU-accelerated indices, achieving sub-millisecond queries over billions of vectors. Libraries implementing product quantization, inverted file indexing, and graph-

based search provide the infrastructure for production embedding retrieval. Among graph-based ANN algorithms, Hierarchical Navigable Small World (HNSW) graphs [16] are well-suited to failure retrieval. HNSW builds a multi-layer graph where upper layers have progressively fewer nodes, letting search descend hierarchically: coarse structure narrows the search region before refinement in the dense bottom layer. This yields logarithmic search complexity and supports incremental insertion, which is important for a growing failure registry. Unlike KD-trees, HNSW maintains high recall in high-dimensional spaces, making it appropriate for the 32-dimensional embedding space used here.

In the failure retrieval context, registry size is typically orders of magnitude smaller than web-scale applications. Failure registries hold thousands to tens of thousands of resolved incidents rather than billions of documents. HNSW therefore provides effectively exact nearest-neighbor search with negligible approximation error at constant-time query cost.

3. Proposed Method

3.1 Failure Record Construction

Failure records are structured features. Each failure episode is represented as a fixed-dimensional feature vector from platform telemetry, capturing four aspects:

Table 1. Feature categories for failure record construction

Category	Example Features	Rationale
Execution behavior	Task duration, stage count, retry count, records processed	Captures the execution profile of the failed run
Resource usage	Peak memory, CPU utilization, shuffle bytes read/written, disk spill	Captures resource pressure patterns
Runtime outcome	Error code, exit status, failure stage index	Encodes the observable failure characteristics
Contextual metadata	Job type indicator, cluster size, data source type	Provides workload context for normalization

Features are extracted from platform system tables and observability APIs at failure time. Categorical features such as error codes are encoded numerically; missing values are imputed using workload-specific defaults.

Consider a representative out-of-memory failure during a shuffle-heavy ETL job. The raw telemetry yields features across the four categories:

Execution behavior: 47 minutes before failing, 12 stages (failing at index 9), 3 retry attempts, ~2.1 billion records processed.

Resource usage: peak executor memory 14.2 GB of 15 GB (94.7%), mean CPU 78%, shuffle 340 GB written and 312 GB read, disk spill 89 GB. The shuffle clearly exceeded memory and spilled heavily before the OOM.

Runtime outcome: error code maps to container-killed (exit 137), task-level failure, failure stage index 9 (shuffle-read).

Contextual metadata: batch ETL on 8 worker nodes at 15 GB each, data source a partitioned Delta Lake table.

After percentile normalization against the historical distribution of batch ETL jobs (Section 3.2), the run sits above the 91st percentile in execution behavior, in the extreme upper tail on resource usage, and carries the specific runtime outcome. A different OOM (producing `GC overhead limit exceeded` rather than a container kill, on a differently sized cluster) would produce a similar normalized vector because the operational pattern (extreme memory pressure, heavy shuffle spill, failure in a shuffle stage) is the same.

3.2 Telemetry Normalization

Comparison demands normalization. Raw telemetry values are not directly comparable across workloads. A percentile-based normalization is applied to each feature relative to the historical distribution of the same workload type.

For each feature f and workload type w, the normalized value is computed as:

$$f\_normalized = (f\_raw - p5(f, w)) / (p95(f, w) - p5(f, w))$$

where p5 and p95 are the 5th and 95th percentiles of feature f for workload type w, computed from a rolling historical window. This normalization preserves interpretability (values near 0 indicate typical behavior; values near 1 indicate extreme behavior) while reducing sensitivity to workload magnitude.

Values below p5 are clipped to 0; values above p95 are clipped to 1. This clipping is intentional for in-distribution analysis but introduces a known limitation under distribution shift (Section 4.3).

Let H(f, w) denote the historical distribution of feature f for workload type w over a rolling T-day window (e.g., T = 90), and Q_alpha(H) the alpha-quantile. The normalization is:

$$f\_norm = clip((f\_raw - Q\_0.05(H(f, w))) / (Q\_0.95(H(f, w)) - Q\_0.05(H(f, w))), 0, 1)$$

Two edge cases warrant explicit treatment. Cold start: when a new workload type is introduced or has fewer than a minimum number of observations (e.g., n_min = 30), per-workload quantiles are unreliable; the system falls back to global quantiles across all workload types and transitions to workload-specific quantiles as history accumulates. Zero-variance features: when Q_0.95 = Q_0.05, the denominator is zero; the normalized value is set to 0.5, and the feature is flagged for potential exclusion for that workload since it carries no discriminative information.

3.3 Contrastive Training

The encoder takes it from there. The normalized failure vectors feed a neural encoder g(x) producing a low-dimensional embedding z = g(x) / ||g(x)|| on the unit hypersphere. The encoder is trained using the supervised contrastive loss [2]:

For each anchor sample i in a batch, the loss sums over all same-class positives p in the batch:

$$L\_i = -(1/|P(i)|) \cdot \Sigma\_\{p \in P(i)\} \log [\exp(z\_i \cdot z\_p / \tau) / \Sigma\_\{a \neq i\} \exp(z\_i \cdot z\_a / \tau)]$$

where P(i) is the set of indices with the same class label as i, tau is a temperature parameter, and the denominator sums over all other samples in the batch.

This loss pulls same-class embeddings together and pushes different-class embeddings apart. Temperature controls the concentration: lower values produce more separated clusters at the cost of harder optimization.

The choice of SupCon over alternative contrastive losses reflects considerations specific to failure embedding. Three candidate losses were evaluated.

Triplet loss [17] operates on (anchor, positive, negative) triplets and enforces a margin between anchor-positive and anchor-negative distances:

$$L\_triplet = max(0, d(z\_a, z\_p) - d(z\_a, z\_n) + margin)$$

Triplet loss has a long history in metric learning but considers only one positive and one negative per anchor, yielding a sparse gradient signal. Its performance is highly sensitive to the mining strategy (hard, semi-hard, or random), and convergence can be slow when most randomly sampled triplets are already satisfied. NT-Xent [1] uses the full batch as negatives but pairs each anchor with a single positive. In the self-supervised setting where positives are augmented views of the same input, this is natural. In the supervised setting where multiple same-class samples exist per batch, NT-Xent discards available positive information.

SupCon [2] generalizes NT-Xent by averaging over all same-class positives. This many-positive formulation exploits all available labels, produces denser gradient updates, and encourages tighter within-class clustering.

Table 2. Comparison of contrastive loss functions for failure embedding

Property	**Triplet Loss**	**NT-Xent**	**SupCon**
Positives per anchor	1	1	All same-class
Negatives per anchor	1	All other in batch	All other in batch
Requires mining strategy	Yes (hard/semi-hard)	No	No
Label utilization	Low (one triplet)	Partial (one positive)	Full (all positives)
Gradient density	Sparse	Moderate	Dense
Simplex convergence guarantee	No	No	Yes [3]
Sensitivity to batch composition	High	Moderate	Low
Suitability for retrieval	Moderate	Good	Strong

The simplex convergence guarantee of SupCon [3] provides a theoretical basis for expecting maximal inter-class separation, which directly benefits nearest-neighbor retrieval. Triplet loss offers no such geometric guarantee, and NT-Xent's single-positive formulation does not yield the same tight within-class clustering SupCon achieves.

Figure 1. Training pipeline for contrastive failure embeddings.

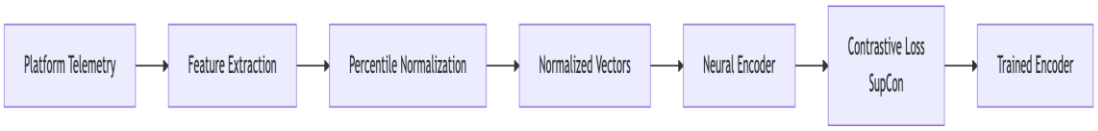

Figure 1 illustrates the end-to-end training pipeline. Raw platform telemetry is transformed into normalized feature vectors, which train the neural encoder via supervised contrastive loss. The trained encoder produces embeddings that position same-class failures close together on the unit hypersphere.

Table 3. Training configuration

Parameter	**Value**	**Rationale**
Encoder architecture	MLP: input dim -> 128 -> 128 -> 32	Sufficient capacity for tabular features
Embedding dimension	32	Exceeds the minimum for class separation on the hypersphere
Temperature (tau)	0.07	Standard value from the SupCon literature
Batch size	256	Large enough for diverse in-batch negatives
Optimizer	Adam, lr=1e-3, weight decay=1e-4	Standard for contrastive training
Epochs	200	With early stopping on validation F1

3.4 Recurrence Detection and Retrieval

Inference is straightforward. After training, the encoder maps new failure records to the embedding space. Recurrence detection and retrieval proceed as follows:

6. 1. Registry population. Embeddings of resolved historical failures are stored in a vector index alongside their metadata and remediation records.
7. 2. Query embedding. When a new failure occurs, its normalized feature vector is passed through the encoder to produce an embedding.
8. 3. Similarity search. The query embedding is compared against the registry using cosine similarity via an approximate nearest neighbor index [15]. The top-k nearest neighbors are returned as candidate historical analogs.
9. 4. Recurrence decision. If the nearest neighbor exceeds a similarity threshold, the failure is flagged as a likely recurrence of the matched historical incident.

Figure 2. Retrieval pipeline for recurrence detection.

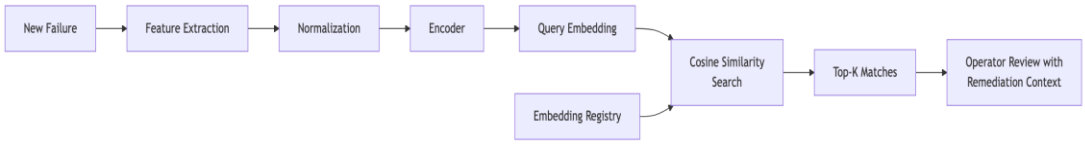

Figure 2 shows the retrieval pipeline at inference. A new failure is encoded into the embedding space and compared against the registry. The top-k nearest neighbors, with their remediation records, are presented to the operator.

This retrieval-based approach supports open-set operation: new failure classes can be added to the registry by embedding and indexing resolved incidents, without retraining the encoder. This distinguishes the contrastive approach from supervised classifiers, which require retraining when the label set changes.

3.5 Training Procedure Details

The training procedure incorporates several design choices intended to stabilize contrastive learning on tabular data and improve generalization.

Learning rate scheduling. The optimizer uses a warmup-then-cosine-decay schedule. During the first W epochs (e.g., W = 10), the learning rate increases linearly from lr_init = 1e-5 to lr_max = 1e-3 to prevent large, poorly directed updates early in training. After warmup, the rate follows cosine annealing:

$$lr(t) = lr\_min + 0.5 \cdot (lr\_max - lr\_min) \cdot (1 + \cos(\pi \cdot (t - W) / (T - W)))$$

where t is the current epoch, T the total epochs, and lr_min a floor (e.g., 1e-6). Cosine decay provides a smooth transition to lower learning rates, shown to improve final representation quality in contrastive learning [18].

Validation strategy. The training set is split into training and validation partitions using stratified sampling to ensure proportional representation of each failure class. At the end of each epoch, the encoder embeds the validation set and a k-nearest-neighbor classifier (k = 5) computes a proxy F1 score. This metric evaluates the embedding space for retrieval without introducing a separate classification head, ensuring the validation signal reflects the intended downstream use.

Early stopping. Training terminates if the validation F1 does not improve for patience consecutive epochs (e.g., 20). The checkpoint with the highest validation F1 is retained. This prevents overfitting, which in contrastive learning can manifest as embeddings overly specialized to training samples rather than general class structure.

Data augmentation considerations. Unlike image-based contrastive learning, the supervised setting allows positive pairs to be defined by labels alone. Light augmentation can still regularize training: additive Gaussian noise (sigma = 0.01) on continuous features after normalization, or feature dropout setting 10% of features to 0.5. Augmentation probability and magnitude are hyperparameters subject to validation-based selection.

3.6 Threshold Selection

The recurrence detection system requires a similarity threshold above which a new failure is declared a likely recurrence. Threshold selection

involves a precision-recall trade-off that depends on the operational context.

At a high threshold (e.g., cosine similarity > 0.95), the system declares recurrence only when the new failure is extremely similar to a registry entry. This produces high precision but low recall, missing genuine recurrences with moderate surface variation. At a low threshold (e.g., > 0.70), the system catches more recurrences but admits false matches that may mislead operators or trigger inappropriate automated remediation.

The optimal operating point depends on downstream consequences. When recurrence detection feeds automated remediation, a high-precision setting is appropriate because applying the wrong remediation may worsen the problem. In an advisory context where candidates are presented for human review, a lower threshold favoring recall is preferable because operators can filter false matches.

The threshold can be selected empirically from a precision-recall curve on a held-out evaluation set. The threshold that maximizes F1 provides a balanced operating point; alternatively, a target precision (e.g., 0.95) can be set and the resulting recall accepted as a design parameter. The threshold may also be set per failure class if classes exhibit tighter or broader embedding distributions. A class with highly consistent failure signatures (e.g., authentication failures with uniform profiles) may support a lower threshold, while a class with high internal variability (e.g., data skew failures whose profiles vary with skew degree) may require a higher threshold.

4. Discussion

4.1 Advantages over Deterministic Fingerprinting

Both approaches have their place. Deterministic fingerprints based on error codes and stack trace hashes provide fast, interpretable matching for exact recurrences. Contrastive embeddings complement this by handling cases where the same root cause produces different surface-level signatures, and the two approaches combine naturally in a dual-mode retrieval system.

Consider two OOM failures from the same recurring pipeline. The first, before a Spark runtime upgrade, produces `java.lang.OutOfMemoryError: Java heap space` at stage 9 with 14.2 GB peak memory and 89 GB spill. The second, after the upgrade, produces a container `OOMKilled` (exit 137) at stage 11, the same logical operation renumbered by the new planner, with 13.8 GB peak memory and 94 GB spill. A fingerprint based on the error string and stage index treats these as unrelated.

In the embedding space, both produce nearly identical normalized feature vectors: extreme memory utilization, heavy shuffle spill, failure at a shuffle-read stage, similar durations and record counts relative to the workload's history. The encoder maps them to nearby points and retrieval surfaces the earlier incident as the top match, letting the operator apply the prior remediation without re-diagnosing.

4.2 Embedding Geometry and Class Separation

The geometry is striking. The supervised contrastive loss encourages class representations to form a regular simplex on the hypersphere [3]. Each failure type occupies a distinct region with maximal angular separation between classes. This geometry is favorable for nearest-neighbor retrieval because it reduces the probability of ambiguous boundary cases.

Embedding dimension must support the number of classes. For k classes, simplex vertices in d dimensions require d >= k - 1. A 32-dimensional space accommodates typical failure taxonomies in enterprise data platforms.

4.3 Normalization and Distribution Shift

The percentile-based normalization is effective for in-distribution workloads but limited under distribution shift. When workloads operate at scales significantly different from the historical training distribution, a large fraction of feature values may saturate at the clipping boundaries, reducing discriminative information in the normalized vector.

The severity of this degradation depends on how many features saturate. In a record with twenty features, if five clip at the upper boundary and three at the lower, 40% of the vector carries only 'extreme high' or 'extreme low' information. For failure types distinguished by specific magnitudes of resource usage, this loss of upper-tail granularity is particularly harmful. The information loss compounds: the encoder receives a feature vector

with reduced effective dimensionality, the embedding is less informative, and cosine similarity becomes less discriminative. Empirically, this manifests as retrieval degradation where top-k matches span multiple failure classes rather than concentrating within the correct class.

This limitation should be acknowledged in deployment planning. Normalization statistics should be updated periodically, and the system should monitor the fraction of saturated features as an indicator of distribution shift. A practical strategy is to trigger a normalization refresh when the rolling-average saturation ratio exceeds a threshold (e.g., 25%).

4.4 Practical Deployment Considerations

Deploying contrastive failure embeddings in a production environment requires attention to several practical factors:

- Labeling. Supervised contrastive learning requires labeled failure data. Labels may come from automated classification (error code mapping), manual annotation, or a combination.
- Encoder updates. As new failure types emerge or existing patterns evolve, the encoder may need periodic retraining.
- Latency. Embedding computation and nearest-neighbor search must be fast enough for real-time or near-real-time retrieval during incident response. GPU-accelerated ANN search [15] can achieve sub-millisecond queries, though enterprise failure registries fit comfortably within CPU-based index structures.
- Integration. The embedding system must connect to the existing observability stack for telemetry input and to incident management tools for output.

4.5 Comparison with Supervised Classification

An alternative is to train a standard supervised classifier (e.g., an MLP with softmax output) to predict the failure class directly. This approach is simple: the model outputs a probability distribution and the argmax class is selected. However, the classification approach and the embedding retrieval approach differ in important respects, summarized in Table 4.

Table 4. Comparison of contrastive embedding retrieval and supervised MLP classification

Property	**Contrastive Embedding + Retrieval**	**Supervised MLP Classifier**
Output	Continuous embedding vector	Probability distribution over fixed label set
Open-set support	Native: novel classes occupy new regions without retraining	None: novel classes require retraining with updated label set
Retraining for new classes	Not required (add to registry)	Required (add output head, retrain)
Similarity granularity	Continuous (cosine similarity)	Discrete (argmax class + confidence)
Retrieval of multiple analogs	Natural (top-k neighbors)	Requires additional design (e.g., class-conditional retrieval)
Interpretability	Requires embedding visualization or feature attribution	Direct via class probabilities
In-distribution accuracy	Comparable to classification (when embeddings are well-separated)	Strong with sufficient labeled data
Deployment complexity	Encoder + vector index + threshold	Classifier + confidence threshold
Confidence calibration	Implicit (similarity score)	Explicit (softmax probabilities, but often miscalibrated)
Robustness to label noise	More robust (noisy labels diluted across many positives)	Less robust (noisy labels directly corrupt decision boundary)

Classification is preferable when the set of failure types is stable, the system requires calibrated probability estimates, and operators expect a single-label answer. Embedding retrieval is preferable when the failure taxonomy is evolving, the system must accommodate novel types without downtime, and operators benefit from a ranked list of similar historical incidents. The two can coexist: a classifier for rapid triage and embeddings for detailed similarity analysis.

4.6 Embedding Space Visualization

Understanding the structure of the learned embedding space reveals encoder quality and expected retrieval behavior. A well-trained contrastive encoder yields an embedding space with characteristic properties. Figure 3 illustrates the resulting geometry: same-class embeddings form tight clusters at distinct simplex vertices, with maximal angular separation between classes.

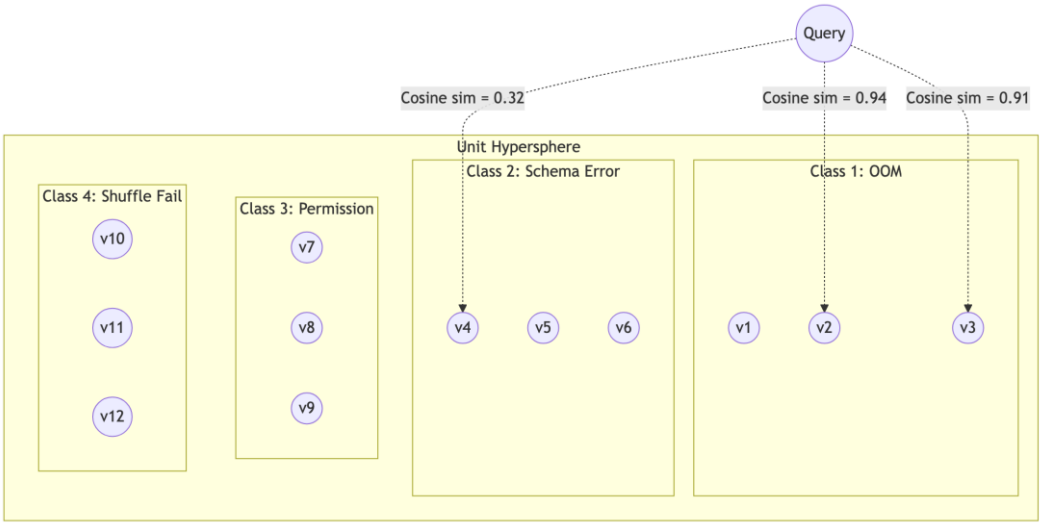

Figure 3. Schematic representation of the contrastive embedding space. Same-class failure embeddings cluster tightly at distinct vertices of a regular simplex on the unit hypersphere [3]. A query embedding for a new OOM failure shows high cosine similarity to the OOM cluster and low similarity to other classes, enabling accurate nearest-neighbor retrieval.

Same-class failures form tight, compact clusters. Projected via t-SNE [19] or UMAP [20], each class appears as a distinct, well-separated cluster. Within-class dispersion reflects natural variability across workloads, but clusters should be substantially tighter than inter-class distances.

Per the simplex convergence property of SupCon [3], class centroids tend toward the vertices of a regular simplex on the unit hypersphere. In the full 32-dimensional space, angular distances between class centroids are approximately equal, though dimensionality-reduction distortion means this is only approximate in two-dimensional visualizations.

Inter-class margins, the angular gaps between the nearest boundaries of adjacent clusters, determine robustness to within-class variability. Larger margins reduce cross-class intrusion during retrieval. The margin can be quantified by computing, for each class, the angular distance between its most peripheral sample and the nearest sample from any other class.

Samples in inter-class regions can indicate partially learned classes, label noise, or genuinely ambiguous failure episodes. A systematic pattern of confusion between two specific classes suggests that the feature set may not contain enough information to distinguish those types, prompting investigation into additional telemetry features.

5. Limitations

1. Scale of evaluation. Evaluation in this study is based on case-based analysis and systematic comparison rather than large-scale controlled experiments. Future work should evaluate on both synthetic and production failure data, reporting retrieval metrics (precision@k, recall@k, MAP) and embedding quality metrics (silhouette score, inter-class angular separation).
2. Synthetic data dependency. Generalization to production distributions remains open. Production data exhibits long-tailed class distributions,

temporal non-stationarity, and correlated feature noise that are difficult to simulate faithfully. Validation across multiple organizations and platform configurations would strengthen confidence in practical applicability.

3. Normalization saturation. The percentile-based normalization is vulnerable to out-of-distribution workloads (Section 4.3). Mitigations include shortening the rolling window, adaptive quantile estimation, or a hybrid scheme combining percentile normalization for in-distribution features with a log-transform fallback for saturating features.

4. Open-set claim. The open-set advantage is a design property, not empirically validated here. Validation should assess whether novel failure types are genuinely separated from known classes in the embedding space or whether they collapse into nearby existing clusters.

5. Single-label assumption. Each failure is assumed to have a single root-cause label; in practice, failures may involve multiple contributing factors. Multi-label extensions of supervised contrastive learning exist but are not explored here.

6. Class imbalance. Production failure distributions are typically long-tailed. Rare classes contribute fewer positive pairs per batch, producing weaker gradient signals. Mitigations include class-balanced sampling, upweighting rare-class pairs, and curriculum learning.

6. Conclusions

The contribution is concrete. This study proposed a contrastive embedding approach for reliability engineering in lakehouse environments. The method constructs failure records from normalized telemetry, trains a neural encoder using supervised contrastive learning, and produces embeddings that capture operational similarity. The resulting space supports recurrence detection, nearest-neighbor retrieval, and similarity-based operational analysis. The contrastive approach extends deterministic fingerprinting with a learned representation layer for cases where exact matching is insufficient. Its open-set design allows new failure classes to be registered without encoder retraining.

The framework addresses the need for a representation that captures semantic similarity of failure episodes rather than relying on surface-level attribute matching. Deterministic approaches remain valuable for speed and interpretability but cannot scale to the combinatorial diversity of failure signatures arising from evolving application code, shifting data distributions, and heterogeneous infrastructure. Contrastive representations offer a path toward broader operational intelligence: failure clustering, trend analysis, and automated triage by embedding similarity. Integration with downstream reasoning systems is a particularly promising direction. Embeddings can serve as a retrieval mechanism for incident summarization and remediation recommendation. Several directions remain open. Future work should extend evaluation to larger-scale production environments and assess robustness under distribution shift and class imbalance. Longitudinal studies would provide evidence on sustainability and retraining cadence.

Declarations

Conflict of Interest

The author declares no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Funding Statement

This research received no external funding.

Ethics Approval

Not applicable. This study involves no human subjects, animal experiments, or sensitive data.

Author Contributions

The author is solely responsible for all aspects of this work, including Conceptualization, Methodology, Formal Analysis, Writing — Original Draft, and Writing — Review and Editing.

Data Availability Statement

The contrastive embedding methodology and training configurations described in this study are fully specified in the manuscript. No proprietary datasets were used. The evaluation is based on synthetic telemetry data generated according to the procedures described in Section 3.

References

[1] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," in Proc. 37th Int. Conf. Machine Learning (ICML), 2020, pp. 1597-1607.

[2] P. Khosla, P. Teterwak, C. Wang et al., "Supervised contrastive learning," in Advances in Neural Information Processing Systems 33 (NeurIPS), 2020, pp. 18661-18673.

[3] F. Graf, C. Hofer, M. Niethammer, and R. Kwitt, "Dissecting supervised contrastive learning," in Proc. 38th Int. Conf. Machine Learning (ICML), 2021, pp. 3821-3830.

[4] D. Bahri, H. Jiang, Y. Tay, and D. Metzler, "SCARF: Self-supervised contrastive learning using random feature corruption," in Proc. 10th Int. Conf. Learning Representations (ICLR), 2022.

[5] J. Yoon, Y. Zhang, J. Jordon, and M. van der Schaar, "VIME: Extending the success of self- and semi-supervised learning to tabular domain," in Advances in Neural Information Processing Systems 33 (NeurIPS), 2020.

[6] T. Ucar, E. Hajiramezanali, and L. Edwards, "SubTab: Subsetting features of tabular data for self-supervised representation learning," in Advances in Neural Information Processing Systems 34 (NeurIPS), 2021.

[7] R. Shwartz-Ziv and A. Armon, "Tabular data: Deep learning is not all you need," *Information Fusion*, vol. 81, pp. 84-90, 2022.

[8] M. Du, F. Li, G. Zheng, and V. Srikumar, "DeepLog: Anomaly detection and diagnosis from system logs through deep learning," in Proc. 2017 ACM SIGSAC Conf. Comput. Commun. Security (CCS), 2017, pp. 1285-1298.

[9] P. He, J. Zhu, Z. Zheng, and M. R. Lyu, "Drain: An online log parsing approach with fixed depth tree," in Proc. 2017 IEEE Int. Conf. Web Services (ICWS), 2017, pp. 33-40.

[10] W. Meng et al., "LogRobust: Robust anomaly detection through template-aware log feature extraction," in Proc. ESEC/FSE, 2019.

[11] X. Zhou, X. Peng, T. Xie, J. Sun, C. Ji, W. Li, and D. Ding, "Fault analysis and debugging of microservice systems: Industrial survey, benchmark system, and empirical study," *IEEE Trans. Software Engineering*, vol. 47, no. 2, pp. 243-260, 2019.

[12] Y. Dang, Q. Lin, and P. Huang, "AIOps: Real-world challenges and research innovations," in *Proc. 41st IEEE/ACM Int. Conf. Software Engineering: Companion (ICSE-Companion)*, 2019, pp. 4-5.

[13] M. Armbrust, A. Ghodsi, R. Xin, and M. Zaharia, "Lakehouse: A new generation of open platforms that unify data warehousing and advanced analytics," in Proc. CIDR, 2021.

[14] M. Armbrust, T. Das, J. Torres et al., "Delta Lake: High-performance ACID table storage over cloud object stores," Proc. VLDB Endowment, vol. 13, no. 12, pp. 3411-3424, 2020.

[15] J. Johnson, M. Douze, and H. Jegou, "Billion-scale similarity search with GPUs," IEEE Trans. Big Data, vol. 7, no. 3, pp. 535-547, 2019.

[16] 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.

[17] F. Schroff, D. Kalenichenko, and J. Philbin, "FaceNet: A unified embedding for face recognition and clustering," in Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR), 2015, pp. 815-823.

[18] I. Loshchilov and F. Hutter, "SGDR: Stochastic gradient descent with warm restarts," in Proc. 5th Int. Conf. Learning Representations (ICLR), 2017.

[19] L. van der Maaten and G. Hinton, "Visualizing data using t-SNE," J. Mach. Learn. Res., vol. 9, pp. 2579-2605, 2008.

[20] L. McInnes, J. Healy, and J. Melville, "UMAP: Uniform manifold approximation and projection for dimension reduction," arXiv preprint arXiv:1802.03426, 2018.