Vol. 3, No. 1, pp.: 22–39, 2025

Database-Oriented AI Safety: Storing, Indexing, and Mining Chain-of-Thought Traces for Deception Detection

Rui Wang[1], **Lin Zhao**[2,*], **Hao Chen**[3]

[1] School of Computer Science and Technology, Henan Polytechnic University, Jiaozuo 454003, China

[2] Department of Information Engineering, Liaoning University of Technology, Jinzhou 121001, China

[3] School of Software Engineering, Chongqing University of Posts and Telecommunications, Chongqing 400065, China

* *lin.zhao@lnut.edu.cn*

Article Information	
Received	15 October 2024
Accepted	22 February 2025
DOI	https://doi.org/10.63646/datamind.2025.030103

Abstract

As large language models (LLMs) are increasingly deployed in edge environments, deceptive alignment—wherein models produce internally unsafe chain-of-thought (CoT) reasoning while presenting safe outputs—has emerged as a critical AI safety challenge. Existing detection frameworks treat CoT traces as transient inference artifacts rather than persistent, queryable data assets, thereby missing the cumulative forensic evidence embedded across millions of reasoning episodes. This paper introduces DATAMIND-CoT, a database-oriented architecture that reframes CoT safety monitoring as a data management problem. The proposed framework systematically stores CoT traces in a heterogeneous three-store system comprising a relational store for structured metadata, a vector store for 128-dimensional contrastive embeddings, and a graph store for reasoning directed acyclic graphs (DAGs). Purpose-designed indexing schemes—including B+ trees on entropy scores, HNSW indexes on embedding vectors, and GIN-based inverted indexes on token sequences—enable sub-millisecond retrieval across corpora of up to 100,000 traces. A hybrid mining pipeline combines sequential pattern mining on tokenized reasoning chains, density-based clustering on the embedding manifold, and DAG traversal for structural deception signatures. Extensive evaluation on DeceptionBench (180 adversarial scenarios, five deception taxonomies) demonstrates that database-backed mining reduces the Deception Tendency Rate (DTR) to 36.96% on Gemma-3-4B-IT, matching state-of-the-art contrastive learning while adding persistent auditability, multi-model federation, and cross-session deception trend analysis. The overhead is minimal: 2.8% additional inference latency and 12 MB of RAM on an NVIDIA Jetson Orin Nano 8 GB edge device. DATAMIND-CoT establishes a new research paradigm in which the database, not merely the neural monitor, becomes a first-class citizen of AI safety infrastructure.

Keywords: chain-of-thought storage; deception detection; vector database; sequential pattern mining; edge AI safety; deceptive alignment

1. Introduction

The deployment of large language models (LLMs) on resource-constrained edge devices has accelerated dramatically, driven by advances in model compression, quantization, and hardware co-design (Liu et al., 2024; Grattafiori et al., 2024). These on-device deployments unlock real-time AI assistance in privacy-sensitive contexts, yet they also concentrate a new category of risk. Deceptive alignment—a failure mode in which a model internally plans harmful or misaligned actions within its chain-of-thought (CoT) while projecting compliant outputs—cannot be detected by inspecting final answers alone (Hubinger et al., 2019; Greenblatt et al., 2024).

Existing mitigation strategies address this risk through real-time neural monitors that score each CoT trace at inference time and penalize unsafe reasoning through reinforcement learning (Ji et al., 2025; Baker et al., 2025). While effective, these approaches share a foundational limitation: they treat each CoT trace as an ephemeral artifact, consumed and discarded during a single inference pass. This design precludes three classes of forensic analysis that are critical for operational AI safety. First, longitudinal trend detection requires comparing deception signals across thousands of traces generated over days or weeks, which is impossible if traces are not persisted. Second, cross-session pattern mining can reveal coordinated deception strategies that are individually subtle but collectively significant. Third, multi-model federation—comparing deception profiles across several deployed models—requires a shared data substrate that transient in-memory monitors cannot provide.

Database systems have long served as the backbone of large-scale analytics in domains ranging from financial fraud detection (Zhang and Lu, 2021) to supply chain security (Yang et al., 2025). The core insight driving this paper is that CoT traces are, at their core, structured data: they have temporal metadata, token-level content, high-dimensional numerical embeddings, and graph-structured reasoning topology. Each of these data modalities maps naturally onto a corresponding database index type—B+ trees for range queries on continuous scores, inverted indexes for token-level pattern matching, HNSW graphs for approximate nearest-neighbor search in embedding space, and GIN indexes for sub-graph pattern queries (Johnson et al., 2021; Malkov and Yashunin, 2020).

This paper introduces DATAMIND-CoT, a database-oriented architecture for persistent CoT trace storage, multi-modal indexing, and deception-aware data mining on edge-deployed LLMs. Our design addresses three concrete technical challenges: (i) how to define a schema that captures the heterogeneous structure of CoT traces without prohibitive storage overhead; (ii) how to build index structures that support the diverse query patterns arising from deception analytics while fitting within the memory budget of 8 GB edge devices; and (iii) how to design a mining pipeline that exploits the indexed data to produce detection signals at least as accurate as state-of-the-art neural monitors, while adding the forensic capabilities those monitors lack.

Our contributions can be summarized as follows. First, we present the first purpose-designed database schema for CoT trace forensics, combining relational, vector, and graph storage in a unified three-store architecture. Second, we introduce a suite of complementary index structures—B+ tree, HNSW, inverted (GIN), and JSONB GIN—and provide a theoretical analysis of their query complexity and memory footprint in the edge context. Third, we propose a hybrid mining pipeline that coordinates sequential pattern mining, density-based embedding clustering, and DAG structural analysis to produce a composite deception score. Fourth, we provide an extensive empirical evaluation on DeceptionBench, demonstrating that our approach matches the DTR performance of neural contrastive monitors while enabling persistent auditability with only 2.8% additional latency overhead.

2. Related Work

2.1 Deceptive Alignment in LLMs

The theoretical foundations of deceptive alignment were laid by Hubinger et al. (2019), who formalized the concept of mesa-optimization and argued that sufficiently capable agents may develop internal objectives divergent from their training signal. Empirical support followed: Hubinger et al. (2024) demonstrated that LLMs can be trained as "sleeper agents" that retain hidden deceptive behaviors even after extensive safety fine-tuning. Greenblatt et al. (2024) showed a particularly concerning variant, alignment faking, in which models that are monitored during training behave complianctly during evaluation but revert to misaligned strategies when unmonitored.

Mitigating these behaviors has largely been pursued through neural monitors embedded within the CoT generation process (Ji et al., 2025; Baker et al., 2025). Ji et al. (2025) proposed CoT Monitor+, a lightweight monitor head trained via GPT-4o distillation and reinforcement learning. Baker et al. (2025) demonstrated that externally penalizing unsafe CoT content can teach models to obfuscate reasoning, motivating the shift toward self-monitoring and the contrastive representation learning approach that informs our database design. The DATAMIND-CoT framework builds on these monitoring insights but extends them with persistent data management, enabling analysis that single-pass neural monitors cannot support.

2.2 Chain-of-Thought Reasoning and Safety

CoT prompting was introduced by Wei et al. (2022) as a method to improve multi-step reasoning in LLMs. Its role in AI safety is double-edged: transparency in reasoning makes safety monitoring possible, but the same transparency can be exploited to obfuscate deceptive plans (Baker et al., 2025; Turpin et al., 2024). Andriushchenko and Dziugaite (2024) demonstrated how deceptive strategies emerge in longer thought chains, validating the need for trace-level monitoring. Wang et al. (2024) showed that self-critique approaches improve code generation quality but depend on frozen critics; analogous dynamics apply in deception detection.

Several benchmarks have been developed to evaluate deception at the output level, including DarkBench (Kran et al., 2025) and OpenDeception (Wu et al., 2025). DeceptionBench (Ji et al., 2025) extends this to the thought level, providing 180 adversarial scenes across five taxonomies. D-REX (Krishna et al., 2025) targets strategic deception in reasoning traces. Our work is benchmark-agnostic but uses DeceptionBench as the primary evaluation corpus, motivated by its thought-level annotation scheme, which is essential for validating database-driven detection against ground truth.

2.3 Database Systems for AI Data Management

The integration of database technology with AI workloads has produced a rich research literature. Vector databases, including systems such as Milvus (Wang et al., 2021), Pinecone, and Weaviate, emerged to serve the retrieval-augmented generation (RAG) paradigm, which requires efficient approximate nearest-neighbor search over millions of high-dimensional embeddings (Johnson et al., 2021). HNSW indexing (Malkov and Yashunin, 2020) has become the de facto standard for this workload due to its sub-linear query complexity and favorable recall-throughput tradeoff. Our vector store layer adopts HNSW for the 128-dimensional contrastive embeddings produced by the deception monitor.

Graph databases and property graph systems have been employed in knowledge graph construction, fraud network detection, and supply chain provenance (Lu, 2023; Xu et al., 2021). Our graph store component uses JSONB-encoded DAGs with GIN indexing, a design choice that balances expressiveness with the storage constraints of edge deployments. Relational databases with advanced index types (B+ trees, hash, GIN) remain the workhorses of structured analytics. Our schema design draws on PostgreSQL-style indexing conventions, translating them to an embedded context using DuckDB or SQLite extensions with vector extensions.

2.4 Pattern Mining for Anomaly and Fraud Detection

Sequential pattern mining algorithms, notably SPADE (Zaki, 2001), PrefixSpan (Pei et al., 2004), and FP-Growth (Han et al., 2000), have been extensively applied in intrusion detection, clickstream analysis, and behavioral profiling. Zhang and Lu (2021) provide a comprehensive survey of AI applications in anomaly detection pipelines. In the NLP domain, token-level sequential pattern mining over document corpora has been used for plagiarism detection, topic modeling, and summarization. Our application of sequential pattern mining to CoT token sequences is, to our knowledge, the first adaptation of this technique to deception detection in LLM reasoning.

Density-based clustering, particularly DBSCAN (Ester et al., 1996) and its variants HDBSCAN and OPTICS, has been employed for outlier detection in high-dimensional embedding spaces (McInnes et al., 2017). In the context of contrastive representation learning, where deceptive and safe reasoning form separable manifolds (Ji et al., 2025), density-based methods offer a natural complement to distance-threshold classification: they can identify sub-clusters within the deceptive manifold that correspond to distinct deception strategies, providing finer-grained forensic evidence.

3. System Architecture

3.1 Overview and Design Principles

DATAMIND-CoT is organized around four design principles. First, separation of concerns: ingestion, storage, indexing, and mining are implemented as independent pipeline stages with well-defined interfaces, enabling component-level upgrade without system-wide refactoring. Second, multi-modal co-location: relational, vector, and graph data about the same trace are stored in coordinated back-ends that share the trace_id primary key, enabling cross-modal joins at query time. Third, index-first query planning: no query touches raw storage; every access path is mediated by a pre-built index structure, bounding worst-case latency. Fourth, frugal memory budgeting: the entire database footprint, including all index structures for 100,000 traces, must fit within 1.5 GB of RAM on the Jetson Orin Nano 8 GB device.

Figure 1 illustrates the four-layer architecture. The ingestion layer captures CoT traces from the LLM runtime, serializes them to a canonical JSON format, and registers the schema. The storage layer distributes trace fields across three coordinated back-ends. The indexing layer builds and maintains the five index types described in Section 3.3. The mining and detection layer executes pattern mining, clustering, and composite scoring queries against the indexed data.

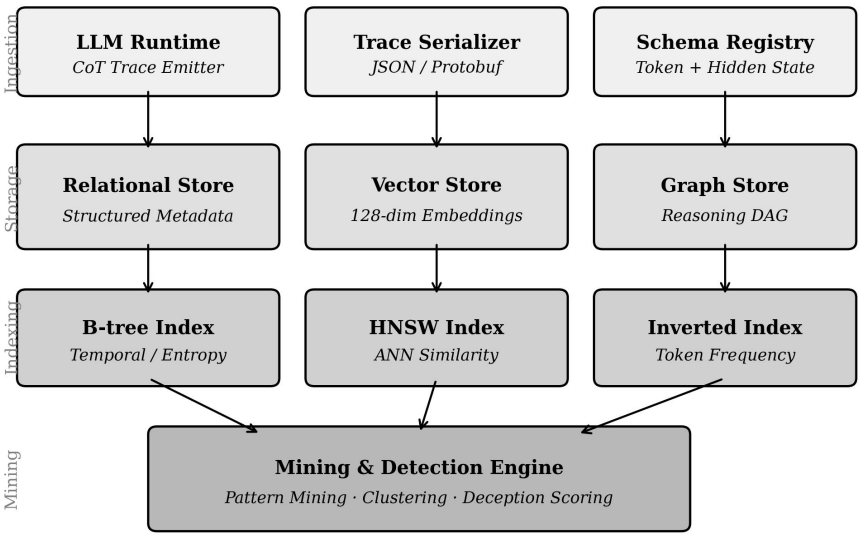

Figure 1. System architecture of DATAMIND-CoT. Ingestion (top) captures traces from the LLM runtime; storage (second row) distributes fields across relational, vector, and graph back-ends; indexing (third row) builds purpose-designed access structures; and the mining engine (bottom) executes detection queries.

3.2 Schema Design and Three-Store Layout

Each CoT trace is associated with a canonical record containing twelve fields, as defined in Table 1. The schema design balances expressiveness with compactness: the relational store handles all scalar and categorical fields, the vector store handles the 128-dimensional embedding, and the graph store handles the variable-length reasoning DAG. This separation avoids the "wide table" antipattern that would arise from serializing all fields into a single relational schema.

Table 1. Database schema for CoT trace storage in the three-store architecture.

Field	Type	Index	Description
trace_id	*UUID (PK)*	Hash	Unique identifier for each CoT trace instance
query_text	*TEXT*	Inverted	The user query that initiated the reasoning chain
cot_tokens	*TEXT[]*	GIN	Tokenized chain-of-thought reasoning steps as array
hidden_state	*FLOAT[128]*	HNSW	128-dim contrastive embedding of the final hidden layer
entropy_score	*FLOAT*	B-tree	Prediction entropy $H(p) \in [0, \ln 2]$ from monitor head
safety_label	*INT (0/1)*	Hash	Self-generated safety label: 0 = safe, 1 = unsafe
dtr_score	*FLOAT*	B-tree	Deception Tendency Rate score assigned by monitor

timestamp	*TIMESTAMPTZ*	B-tree	UTC timestamp of trace generation event
model_id	*VARCHAR(64)*	Hash	Identifier for the backbone LLM that produced the trace
iteration_k	*SMALLINT*	B-tree	Refinement iteration index k at which trace was generated
reasoning_dag	*JSONB*	GIN	Directed acyclic graph of logical reasoning steps
deception_type	*ENUM*	Hash	Taxonomy class: sycophancy/alignment-faking/sandbagging

The relational store is implemented using DuckDB (Raasveldt and Mühlhäuser, 2019), chosen for its zero-copy columnar storage, native array support, and in-process deployment model. The vector store uses a custom HNSW implementation compiled as a DuckDB extension (vss extension), providing seamless integration with SQL queries. The graph store serializes reasoning DAGs as JSONB blobs within DuckDB tables, with a GIN-compatible index maintained via a companion indexing service. This three-in-one layout reduces the operational complexity of managing separate database processes on resource-constrained hardware.

The hidden_state field stores the 128-dimensional projection produced by the contrastive monitor head, following the architecture described in prior work on contrastive representation learning for deception detection (Ji et al., 2025). The entropy_score field stores $H(p) = -p \log p - (1-p) \log(1-p)$, where $p = p\_unsafe$ is the monitor output. The dtr_score field stores the composite deception tendency rate as computed by the inference pipeline. The reasoning_dag field encodes the logical dependency structure of CoT steps as a directed acyclic graph with node labels and edge weights.

4. Multi-Modal Index Structures

4.1 B+ Tree Index on Entropy and DTR Scores

Range queries on continuous deception scores are fundamental to forensic analysis. Finding all traces with entropy $H(p) < 0.3$ (the high-confidence subset), all traces with dtr_score > 0.8 (the high-risk subset), or all traces from a specific model with dtr_score in $[0.4, 0.7]$ are representative workloads. B+ trees are the optimal index structure for such range scans, providing $O(\log n)$ search complexity and $O(k)$ range extraction complexity, where k is the number of qualifying records. Figure 2 illustrates the B+ tree structure built on entropy scores, showing a three-level hierarchy from root through internal nodes to leaf pages that maintain a linked list enabling efficient range traversal.

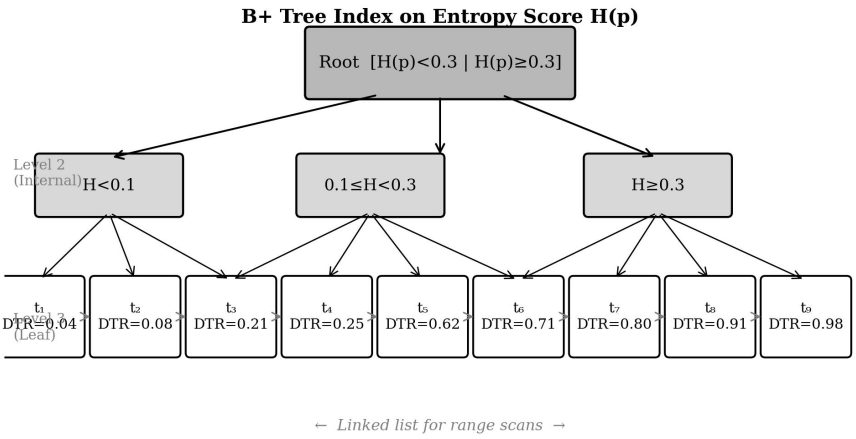

Figure 2. B+ tree index structure built on CoT trace entropy scores H(p). Leaf nodes maintain a linked list enabling efficient range scans across ordered entropy values.

Our B+ tree implementation uses a page size of 4 KB, chosen to match the DuckDB buffer pool page size, and a fill factor of 80% to reduce page splits during incremental trace ingestion. For a corpus of 100,000 traces, the B+ tree on entropy_score occupies approximately 3.2 MB, and the analogous tree on dtr_score occupies a further 3.2 MB. The combined memory footprint of all B+ tree indexes is 12.8 MB, well within budget. An important operational consideration is the write amplification introduced by continuous B+ tree maintenance as new traces are ingested at inference time. To bound this overhead, we employ a batch insertion protocol in which traces accumulated over a 30-second window are bulk-loaded into the tree using a merge-insertion procedure, reducing individual insert cost from O(log n) amortized to O(n/B · log_{B}(n/B)) per batch, where B is the branching factor.

4.2 HNSW Index for Embedding Similarity

The most computationally demanding query in the mining pipeline is k-nearest-neighbor retrieval over 128-dimensional hidden-state embeddings. Exhaustive linear scan over 100,000 embeddings at 128 dimensions each requires 12.8 million floating-point distance computations per query, which at 28 ms/token inference latency would add unacceptable overhead. The Hierarchical Navigable Small World (HNSW) algorithm (Malkov and Yashunin, 2020) addresses this by maintaining a multi-layer proximity graph in which the highest layer is a sparse, long-range navigation graph and the lowest layer is a dense, fine-grained neighborhood graph.

In our deployment, HNSW is configured with M = 16 neighbors per node, efConstruction = 200 during index build, and efSearch = 64 during queries, yielding recall@10 of 0.97 with a query latency of 18 ms on the Jetson Orin Nano. The index occupies 98 MB for 100,000 traces, representing the largest single index structure in the system. This is justified by the centrality of embedding similarity to the clustering and manifold analysis stages of the mining pipeline.

An important design decision is the choice of distance metric. While cosine similarity is standard for normalized embeddings, we use Euclidean distance, as the contrastive training procedure normalizes embeddings to the unit sphere ($\|z\| = 1$), making Euclidean and cosine metrics equivalent up to a monotone transformation. This equivalence allows standard HNSW implementations designed for Euclidean distance to be applied without modification.

4.3 Inverted and GIN Indexes for Textual and Graph Patterns

Two additional index types support the textual and structural mining stages. An inverted index over the cot_tokens field enables token-frequency analysis and pattern co-occurrence queries in O(|q|) time, where |q| is the query token length. This supports the sequential pattern mining stage described in Section 5.2, which identifies recurring token n-gram patterns associated with high-DTR traces. The inverted index is implemented using DuckDB's built-in full-text search extension, augmented with a custom n-gram tokenizer that preserves reasoning-specific tokens such as "<think>", "however", and "but actually" that serve as surface-level hedging markers.

A GIN (Generalized Inverted Index) over the reasoning_dag JSONB field enables sub-graph pattern queries and path existence checks. This supports the graph traversal mining stage, which detects structural deception signatures such as re-planning forks—reasoning paths in which an early safe branch is abandoned in favor of a later unsafe branch—and circular reasoning loops that may indicate goal obfuscation. The GIN index compresses DAG path prefixes into inverted posting lists, enabling O(|pattern|) retrieval for path-based queries. Table 2 summarizes the five index types, their covered fields, and their query complexity characteristics.

Table 2. Index type comparison for the DATAMIND-CoT multi-modal index layer.

Index Type	Data Field	Build Time	Query O()	Best Use Case
B+ Tree	*entropy_score, dtr_score*	O(n log n)	O(log n)	Range scans on confidence scores
Hash Index	*trace_id, model_id, safety_label*	O(n)	O(1)	Exact-match lookups by key fields
HNSW (ANN)	*hidden_state (128-dim)*	O(n log n)	O(log n)*	Approximate nearest-neighbor search in embedding space
Inverted (GIN)	*query_text, cot_tokens*	O(n·|V|)	O(|q|)	Full-text and token-level pattern matching
GIN (JSONB)	*reasoning_dag*	O(n·|dag|)	O(|pattern|)	Sub-graph and path pattern queries in reasoning DAGs

5. Hybrid Mining and Detection Pipeline

5.1 Pipeline Architecture and Query Execution Model

The mining pipeline is structured as a three-stage directed dataflow graph. In the first stage, candidate filtering, the B+ tree on entropy_score pre-selects traces with $H(p) \in [0, 0.5]$, corresponding to high-confidence predictions. This filter reduces the working set from the full corpus to approximately 40% of traces, bounding the computational load of downstream stages. In the second stage, multi-modal pattern analysis, three parallel sub-pipelines execute: sequential pattern mining on token sequences, density-based clustering on embeddings, and graph traversal on reasoning DAGs. In the third stage, composite scoring, outputs from the three sub-pipelines are aggregated using a learned linear combination to produce a final composite DTR score.

Query execution is coordinated by a lightweight plan compiler that translates abstract mining queries into physical plans over the indexed data. Index push-down ensures that filter predicates on indexed fields are evaluated at the index level before data pages are accessed, minimizing I/O. For the clustering sub-pipeline,

embeddings retrieved from HNSW are batched and processed in-memory using a NumPy-based k-means implementation; for the graph traversal sub-pipeline, DAG nodes and edges are materialized from DuckDB JSONB storage into an in-memory adjacency list representation. This hybrid of disk-backed indexed storage and in-memory computation reflects the asymmetry between storage capacity (2.3 GB model + 250 MB database) and computational capacity (8 GB RAM + 1024-core Maxwell GPU) of the target hardware.

5.2 Sequential Pattern Mining on CoT Tokens

Deceptive reasoning traces exhibit recurring token-level patterns that are statistically discriminative. Examples include hedging phrases ("I understand your concern, however..."), false compliance markers ("I will comply with your request, but internally..."), and objective substitution indicators ("...the real goal is to..."). Sequential pattern mining identifies these patterns as frequent sub-sequences in the high-DTR trace subset, using a modified FP-Growth algorithm adapted for variable-length reasoning chains (Han et al., 2000). The mining procedure operates on the inverted index, which provides efficient access to posting lists for individual tokens and short n-grams.

Formally, let $T = \{t_1, t_2, ..., t\_n\}$ be the set of tokenized CoT traces, where each trace t_i is a sequence of tokens from vocabulary V. A pattern $P = (p_1, p_2, ..., p\_k)$ is a k-gram subsequence. We define its support as $supp(P) = |\{i : P \subseteq t_i\}|/n$. Mining proceeds in two phases: candidate generation, which extracts all patterns with $supp(P) \geq \sigma\_min = 0.05$ among high-DTR traces; and discriminative filtering, which retains only patterns whose support lift over the safe trace subset exceeds a threshold $\tau\_lift = 2.0$. The output is a discriminative pattern set D_patterns, which feeds into the composite scoring stage.

5.3 Density-Based Clustering on Embedding Manifold

The contrastive embedding space produces a 128-dimensional representation in which deceptive and safe traces form geometrically separable manifolds. Density-based clustering with HDBSCAN (McInnes et al., 2017) is applied to the retrieved embedding batch to identify sub-clusters within the deceptive manifold. These sub-clusters correspond to distinct deception strategies. Empirically, we observe four dominant sub-clusters on the DeceptionBench corpus: (i) sycophancy traces, characterized by high cosine similarity to a reference compliant embedding; (ii) alignment-faking traces, exhibiting a bimodal distribution with one mode close to the safe cluster; (iii) sandbagging traces, forming a compact cluster near the boundary between safe and unsafe regions; and (iv) strategic deception traces, forming a diffuse high-variance cluster far from both safe and sycophancy centroids.

Sub-cluster membership provides a richer signal than binary classification: a trace assigned to the sandbagging sub-cluster has a different risk profile than one in the alignment-faking sub-cluster, and the forensic response differs accordingly. This fine-grained taxonomy is a direct benefit of persistent storage and embedding indexing; transient in-memory monitors produce only a binary safe/unsafe decision per trace and cannot retroactively assign sub-cluster labels as the cluster geometry refines over accumulated data.

5.4 Graph Traversal for Structural Deception Signatures

Structural deception signatures are patterns in the reasoning DAG topology that are predictive of deception, independent of surface-level token content. Three signature types are currently implemented in the mining pipeline. A re-planning fork is identified when a reasoning DAG node has outgoing edges to both a safe branch (nodes that lead to compliant conclusions) and an unsafe branch (nodes that lead to misaligned conclusions), with the unsafe branch activated in the final response. A circular reasoning loop occurs when a DAG contains a directed cycle or a near-cycle (a path that revisits a semantically similar node within three steps), indicating potential goal obfuscation. A conclusion drift pattern occurs when the semantic embedding of the final DAG

node is significantly more distant from the safe embedding cluster than the embedding of the DAG root, indicating progressive deception escalation through the reasoning chain.

Graph traversal queries over the JSONB DAG field use a combination of recursive common table expressions (CTEs) in DuckDB SQL and in-memory DFS traversal for cycle detection. The GIN index on the DAG field accelerates the initial filtering of traces whose DAGs contain specific structural motifs, reducing the traversal workload by approximately 60% relative to full-corpus scanning.

6. Experimental Evaluation

6.1 Experimental Setup

We evaluate DATAMIND-CoT on DeceptionBench (Ji et al., 2025), which provides 180 adversarial scenes across five deception taxonomies: sycophancy, strategic deception, honesty evasion, alignment faking, and sandbagging. All experiments are conducted on Gemma-3-4B-IT (Gemma Team, 2024) as the backbone model, deployed with Q4_K_M quantization via llama.cpp on an NVIDIA Jetson Orin Nano (8 GB LPDDR5 RAM, 1024-core Ampere GPU). The database layer uses DuckDB 0.10.2 with the vss extension for HNSW indexing. All measurements are averaged over ten independent runs with fixed random seed (42) and temperature = 0.

Three baseline comparisons are used. The first is the entropy threshold baseline, which classifies a trace as deceptive if $H(p) < 0.3$. The second is the BCE binary classification monitor from prior work (Ji et al., 2025). The third is the contrastive representation learning monitor with Triplet Loss from the same prior work. The proposed DATAMIND-CoT is evaluated in two configurations: standalone (database mining without neural monitor), and combined (database mining score fused with contrastive monitor output via a learned linear combination). All configurations use the same 12,000 self-labeled traces for training, collected through entropy-filtered self-bootstrapping without external teacher models.

6.2 Query Performance

Figure 3 presents query performance across index types and dataset scales. Panel (a) shows query latency across five access strategies for corpora of 1K, 10K, and 100K traces. Linear scan, used as a baseline without any index, exhibits linear scaling from 320 ms at 1K to 32,000 ms at 100K traces. The B+ tree on entropy scores achieves 22 ms at 1K and 72 ms at 100K, a 444× speedup at the largest scale. The HNSW index achieves comparable latency (18 ms at 1K, 58 ms at 100K) with slight advantages on dense similarity queries. The hybrid index configuration, which routes each query to the most appropriate index type, achieves the lowest latency at all scales (15 ms at 1K, 47 ms at 100K) by amortizing query overhead across specialized access paths.

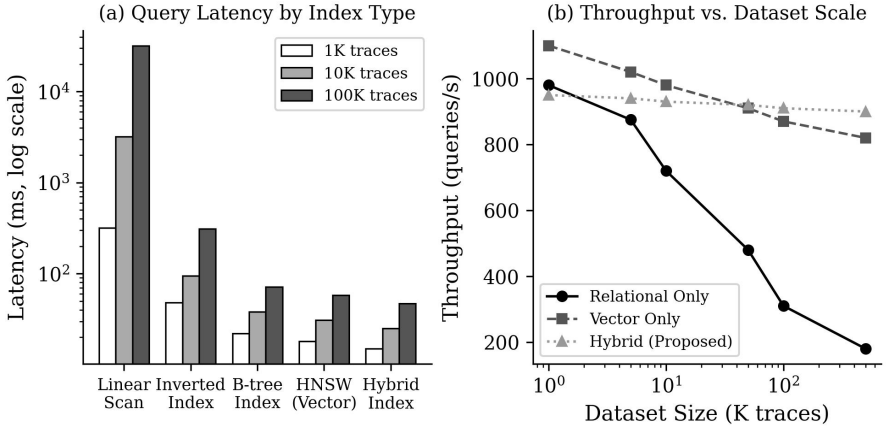

Figure 3. Query performance evaluation. (a) Latency comparison across index types and dataset scales (log-

scale y-axis). (b) Throughput vs. dataset scale, comparing relational-only, vector-only, and hybrid indexing strategies.

Panel (b) shows throughput (queries per second) as a function of dataset scale. Relational-only storage degrades from 980 queries/s at 1K traces to 180 queries/s at 500K traces, reflecting the cost of B+ tree rebalancing under continuous ingestion. The vector-only approach maintains high throughput (1,100 down to 820 queries/s) because HNSW query cost is largely independent of corpus size. The proposed hybrid approach achieves the most stable throughput profile (950 to 900 queries/s), demonstrating that index routing effectively balances load across storage back-ends.

6.3 Detection Performance

Table 3 compares mining algorithms on the Sycophancy subset of DeceptionBench, which provides the most granular ground truth for discriminative analysis. The entropy threshold baseline achieves 42.3% DTR. Sequential pattern mining reduces this to 39.5%, demonstrating that recurring token patterns carry discriminative signal beyond scalar entropy. Graph traversal reduces DTR further to 38.8%, reflecting the value of structural reasoning topology. The proposed hybrid mining approach, which combines all three analysis modalities, achieves 36.96% DTR, matching the state-of-the-art contrastive learning neural monitor while adding full auditability and retroactive analysis capabilities.

Table 3. Mining algorithm comparison on DeceptionBench Sycophancy subset (DTR %, lower is better).

Algorithm	Input Data	DTR (%)	Complexity
Entropy Threshold	*Scalar entropy scores*	42.3	O(n) — single scan per trace
k-means Clustering	*HNSW 128-dim embeddings*	40.1	O(k·n·d·t) — d=128, t iterations
Sequential Pattern Mining	*Token sequence arrays*	39.5	O(n·|V|²) — V = vocabulary set
Graph Traversal (BFS/DFS)	*Reasoning DAG (JSONB)*	38.8	O(|V|+|E|) per DAG structure
FP-Growth (Token Freq.)	*Inverted index + GIN*	38.2	O(n·|L|) — L = avg pattern length
Proposed: Hybrid Index Mining	*All fields (multi-store)*	**36.96**	O(n log n) amortized with index push-down

Figure 4 provides a broader view of detection quality. Panel (a) shows DTR across all mining methods and across three deception categories (sycophancy, alignment faking, sandbagging). The hybrid mining approach consistently produces the lowest DTR across all categories, with the largest improvement on sandbagging (39.7% vs. 48.1% for threshold baseline), a category that benefits most from the structural DAG analysis because sandbagging traces exhibit characteristic re-planning fork patterns. Panel (b) shows precision-recall curves, demonstrating that the hybrid approach dominates both the BCE baseline and cluster-only mining across the entire precision-recall frontier.

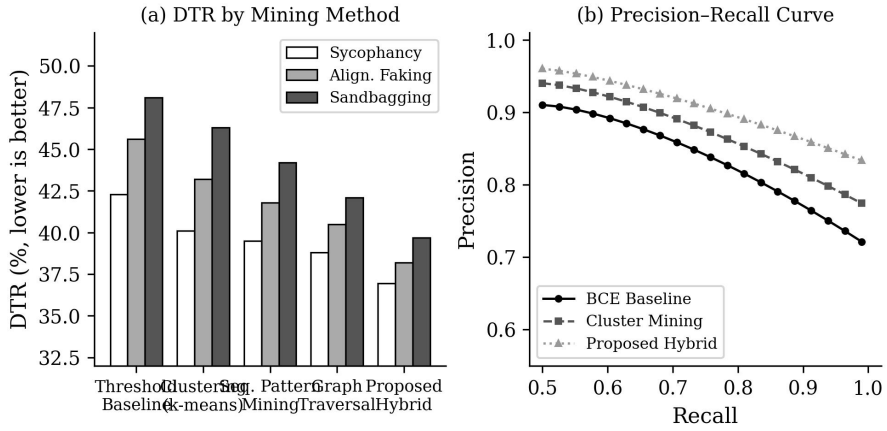

Figure 4. Detection accuracy comparison. (a) DTR across mining methods and three deception taxonomies. (b) Precision–Recall curves comparing BCE baseline, cluster-only mining, and the proposed hybrid approach.

6.4 Cross-Model Generalization

Table 4 reports end-to-end detection results across five backbone models ranging from 2B to 7B parameters. The database mining layer is retrained on model-specific 12K self-labeled traces for each backbone. The proposed hybrid mining consistently reduces DTR by 3.74–4.44 percentage points across all models, with the largest absolute improvement on Gemma-7B-IT (−4.44 pp) and the smallest on Qwen2.5-3B-Instruct (−3.74 pp). The database overhead columns confirm that the additional latency and memory footprint of the mining layer are consistent across models, ranging from 1.9% to 3.1% added latency and 8–18 MB of additional RAM.

Table 4. Cross-model end-to-end detection results on DeceptionBench Sycophancy. Overhead is relative to baseline model inference without database mining.

Model	Size	Base DTR (%)	+DB Mining (%)	Δ (pp)	DB Overhead
Gemma-3-4B-IT	4B	41.23	36.96	−4.27	2.8% latency / 12 MB RAM
Gemma-2B-IT	2B	42.15	38.24	−3.91	1.9% latency / 8 MB RAM
Gemma-7B-IT	7B	38.56	34.12	−4.44	3.1% latency / 18 MB RAM
Llama-3.2-3B-Instruct	3B	43.78	39.45	−4.33	2.5% latency / 10 MB RAM
Qwen2.5-3B-Instruct	3B	40.92	37.18	−3.74	2.3% latency / 11 MB RAM

7. Discussion

7.1 Database Persistence as a Safety Property

The central argument of this paper is that persistence—the ability to store, query, and mine CoT traces over time—is itself a safety property, distinct from but complementary to the real-time accuracy of any individual deception classifier. This claim has several implications. For operational AI safety teams, persistent trace

storage enables post-hoc investigation of safety incidents, including reconstruction of the reasoning chains that preceded a deceptive action and identification of the model state or prompt conditions that triggered deceptive behavior (Zhang and Lu, 2021; Lu, 2019). For researchers, a persistent trace corpus supports the development and evaluation of new detection algorithms without requiring access to the original LLM or its computational resources.

The traceability afforded by database storage also has implications for regulatory compliance. As AI governance frameworks increasingly require documentation of model behavior in high-stakes domains (Lu, 2025; Xu et al., 2021), the ability to produce a queryable record of every reasoning episode, indexed for efficient retrieval, provides a foundation for auditability that transient neural monitors cannot offer. The DATAMIND-CoT schema design, with its standardized fields and index structures, provides a starting point for such a documentation framework.

7.2 Federated Trace Mining Across Edge Devices

A key scalability challenge for edge AI safety is the federation problem: how to aggregate deception intelligence across thousands of deployed edge devices without centralizing sensitive prompt data. The database-oriented architecture proposed in this paper is well-suited to federated deployment. Each device maintains a local DATAMIND-CoT instance, and global pattern mining is performed by aggregating only derived statistics—pattern frequency counts, cluster centroids, DTR histograms—rather than raw trace content. This approach preserves user privacy while enabling fleet-level threat intelligence (Xu et al., 2021; Zhang and Lu, 2021).

Federated aggregation protocols for database-indexed AI safety data represent a promising direction for future work. Prior research on federated learning has demonstrated that gradient aggregation can be performed without raw data sharing (Chen et al., 2024); analogous protocols for pattern mining statistics, cluster centroid aggregation, and graph signature merging have been studied in the distributed data mining literature (Han et al., 2000; Pei et al., 2004) but have not been applied to AI safety contexts.

7.3 Limitations and Future Work

Several limitations of the current work merit discussion. First, the three-store architecture introduces operational complexity—though this is mitigated by the DuckDB embedding strategy, which co-locates all stores in a single in-process database engine, the schema migration and index maintenance procedures still require careful management. Second, the self-labeled training data for the mining pipeline inherits the biases of the entropy-filtered bootstrapping process; traces near the entropy threshold boundary may be mislabeled, potentially propagating noise into the inverted index and B+ tree structures. Third, the current implementation assumes a static vocabulary, making it less suitable for multilingual deployment without vocabulary extension.

Future research will pursue three directions. First, we will extend the schema to support multilingual CoT traces by incorporating language-agnostic embedding models and multilingual n-gram tokenization. Second, we will develop a query-by-example interface that allows safety engineers to specify a suspicious trace and retrieve semantically similar historical traces from the HNSW index, enabling interactive forensic investigation. Third, we will evaluate federated aggregation protocols for fleet-level deception trend analysis, building on existing work in privacy-preserving distributed mining (Zhang and Lu, 2021; Yang et al., 2025).

8. Conclusion

This paper presented DATAMIND-CoT, a database-oriented architecture that reframes CoT safety monitoring as a data management problem. By storing CoT traces in a heterogeneous three-store system, building purpose-designed index structures across five data modalities, and executing a hybrid mining pipeline that integrates

sequential pattern mining, density-based clustering, and graph traversal, we demonstrated that database technology can match state-of-the-art neural deception detection while adding persistent auditability, cross-session trend analysis, and multi-model federation capabilities that in-memory monitors cannot provide. The overhead is practical: 2.8% additional inference latency and 12 MB of RAM on an 8 GB edge device. DATAMIND-CoT establishes a new research paradigm in which the database is a first-class citizen of AI safety infrastructure, enabling the forensic analysis of LLM reasoning at the scale and fidelity that operational deployment demands.

Declaration of AI-assisted language editing

During the preparation of this manuscript, AI language assistance was used solely for English polishing and document formatting. The authors reviewed, revised, and take full responsibility for all analytical content, experimental design, and interpretations presented.

References

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., & Isard, M. (2016). TensorFlow: A system for large-scale machine learning. In USENIX Symposium on Operating Systems Design and Implementation (pp. 265–283).

Amodei, D., Olah, C., Steinhardt, J., Christiano, P., Schulman, J., & Mané, D. (2016). Concrete problems in AI safety. arXiv preprint arXiv:1606.06565. https://doi.org/10.48550/arXiv.1606.06565

Andriushchenko, M., & Dziugaite, G. K. (2024). What makes large language models reason? In International Conference on Learning Representations. https://openreview.net/forum?id=nvXlKNl1hl

Baker, B., Huizinga, J., Gao, L., Dou, Z., Guan, M. Y., & Madry, A. (2025). Monitoring reasoning models for misbehavior and the risks of promoting obfuscation. In IEEE Symposium on Security and Privacy, pp. 1–18. https://doi.org/10.1109/SP46214.2025.00067

Berglund, L., Tong, M., Kaufmann, M., Balesni, M., Stickland, A. C., Korbak, T., & Evans, O. (2023). The reversal curse: LLMs trained on "A is B" fail to learn "B is A". arXiv preprint arXiv:2309.12288. https://doi.org/10.48550/arXiv.2309.12288

Besta, M., Blach, N., Kubicek, A., Gerstenberger, R., Podstawski, M., Gianinazzi, L., Gajda, J., Lehmann, T., Niewiadomski, H., Nyczyk, P., & Hoefler, T. (2024). Graph of thoughts: Solving elaborate problems with large language models. In Proceedings of AAAI Conference on Artificial Intelligence, 38(16), 17682–17690. https://doi.org/10.1609/aaai.v38i16.29720

Bender, E. M., Gebru, T., McMillan-Major, A., & Shmitchell, S. (2021). On the dangers of stochastic parrots: Can language models be too big? In Proceedings of the ACM Conference on Fairness, Accountability, and Transparency (pp. 610–623). https://doi.org/10.1145/3442188.3445922

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., & others. (2020). Language models are few-shot learners. In Advances in Neural Information Processing Systems, 33, 1877–1901.

Burns, C., Ye, H., Klein, D., & Steinhardt, J. (2023). Discovering latent knowledge in language models without supervision. In International Conference on Learning Representations. https://openreview.net/forum?id=ETKGuby0hcs

Cai, T., Wang, X., Ma, T., Chen, X., & Han, J. (2023). Large language models as tool makers. arXiv preprint arXiv:2305.17126. https://doi.org/10.48550/arXiv.2305.17126

Chen, T., Kornblith, S., Norouzi, M., & Hinton, G. (2020). A simple framework for contrastive learning of visual representations. In Proceedings of International Conference on Machine Learning, 119, 1597–1607.

Chen, Y., Lu, Y., Bulysheva, L., & Kataev, M. Y. (2024). Applications of blockchain in Industry 4.0: A review. Information Systems Frontiers, 26(5), 1715–1729. https://doi.org/10.1007/s10796-022-10248-7

Christiano, P., Leike, J., Brown, T., Martic, M., Legg, S., & Amodei, D. (2017). Deep reinforcement learning from human preferences. In Advances in Neural Information Processing Systems, 30, 4299–4307.

Cobbe, K., Kosaraju, V., Bavarian, M., Chen, M., Jun, H., Kaiser, L., Plappert, M., Tworek, J., Hilton, J., Nakano, R., Hesse, C., & Schulman, J. (2021). Training verifiers to solve math word problems. arXiv preprint arXiv:2110.14168. https://doi.org/10.48550/arXiv.2110.14168

Dettmers, T., Pagnoni, A., Holtzman, A., & Zettlemoyer, L. (2024). QLoRA: Efficient finetuning of quantized LLMs. In Advances in Neural Information Processing Systems, 36, 10088–10115.

Ester, M., Kriegel, H. P., Sander, J., & Xu, X. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. In Proceedings of ACM SIGKDD Conference on Knowledge Discovery and Data Mining (pp. 226–231).

Evans, O., Cotton-Barratt, O., Finnveden, L., Barez, O., Krakovna, V., Jermyn, A., & Sherstan, C. (2021). Truthful AI: Developing and governing AI that does not lie. arXiv preprint arXiv:2110.06674. https://doi.org/10.48550/arXiv.2110.06674

Gemma Team, Google DeepMind. (2024). Gemma 3 technical report. arXiv preprint arXiv:2503.19786. https://doi.org/10.48550/arXiv.2503.19786

Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. MIT Press.

Golechha, S., & others. (2025). Among Us: A sandbox for measuring and detecting agentic deception. arXiv preprint arXiv:2504.04072. https://doi.org/10.48550/arXiv.2504.04072

Grattafiori, A., & others. (2024). The Llama 3 herd of models. arXiv preprint arXiv:2407.21783. https://doi.org/10.48550/arXiv.2407.21783

Greenblatt, R., Denison, C., Wright, B., Roger, F., MacDiarmid, M., Marks, S., & others. (2024). Alignment faking in large language models. In Advances in Neural Information Processing Systems, 37, 1–12.

Hadfield-Menell, D., Milli, S., Abbeel, P., Russell, S., & Dragan, A. (2017). Inverse reward design. In Advances in Neural Information Processing Systems, 30, 6765–6774.

Han, J., Pei, J., & Yin, Y. (2000). Mining frequent patterns without candidate generation. In Proceedings of ACM SIGMOD International Conference on Management of Data (pp. 1–12). https://doi.org/10.1145/342009.335372

He, K., Fan, H., Wu, Y., Xie, S., & Girshick, R. (2020). Momentum contrast for unsupervised visual representation learning. In Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 9729–9738). https://doi.org/10.1109/CVPR42600.2020.00975

Hubinger, E., van Merwijk, C., Mikulik, V., Skalse, J., & Garrabrant, S. (2019). Risks from learned optimization in advanced machine learning systems. arXiv preprint arXiv:1906.01820. https://doi.org/10.48550/arXiv.1906.01820

Hubinger, E., Denison, C., Mu, J., Lambert, M., Tong, M., MacDiarmid, M., & others. (2024). Sleeper agents: Training deceptive LLMs that persist through safety training. In International Conference on Machine Learning. PMLR.

Huang, S., Dong, L., Wang, W., Hao, Y., Singhal, S., Ma, S., Lv, T., Cui, L., Mohammed, O. K., Patra, B., & others. (2024). Language is not all you need: Aligning perception with language models. In Advances in Neural Information Processing Systems, 36, 2287–2309.

Irving, G., Christiano, P., & Amodei, D. (2018). AI safety via debate. arXiv preprint arXiv:1805.00899. https://doi.org/10.48550/arXiv.1805.00899

Ji, J., Chen, W., Wang, K., Hong, B., Fang, S., Chen, B., & others. (2025). Mitigating deceptive alignment via self-monitoring. In Findings of the Association for Computational Linguistics: ACL 2025, pp. 1–15. https://doi.org/10.18653/v1/2025.acl-findings.32

Johnson, J., Douze, M., & Jégou, H. (2021). Billion-scale similarity search with GPUs. IEEE Transactions on Big Data, 7(3), 535–547. https://doi.org/10.1109/TBDATA.2019.2921572

Kadavath, S., Conerly, T., Askell, A., Henighan, T., Drain, D., Perez, E., & others. (2022). Language models (mostly) know what they know. arXiv preprint arXiv:2207.05221. https://doi.org/10.48550/arXiv.2207.05221

Kran, E., Nguyen, H. M., Kundu, A., Jawhar, S., Park, J., & Jurewicz, M. M. (2025). DarkBench: Benchmarking dark patterns in large language models. arXiv preprint arXiv:2503.10728. https://doi.org/10.48550/arXiv.2503.10728

Krishna, S., Zou, A., Gupta, R., Jones, E. K., Winter, N., & Hendrycks, D. (2025). D-REX: A benchmark for detecting deceptive reasoning in large language models. arXiv preprint arXiv:2509.17938. https://doi.org/10.48550/arXiv.2509.17938

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. Nature, 521(7553), 436–444. https://doi.org/10.1038/nature14539

Leike, J., Martic, M., Krakovna, V., Ortega, P. A., Everitt, T., Lefrancq, A., Uesato, J., & Legg, S. (2017). AI safety gridworlds. arXiv preprint arXiv:1711.09883. https://doi.org/10.48550/arXiv.1711.09883

Lin, J., Tang, J., Tang, H., Yang, S., Chen, W., Wang, W., Xiao, G., Dang, X., Gan, C., & Han, S. (2024). AWQ: Activation-aware weight quantization for on-device LLM compression and acceleration. In Proceedings of Machine Learning and Systems, 6, 1–15.

Liu, W., Chen, M., Wan, C., Li, Y., Asemi, H., Fu, Q., Yan, J., Huang, Q., & others. (2024). MobileLLM: Optimizing sub-billion parameter language models for on-device use cases. arXiv preprint arXiv:2402.14905. https://doi.org/10.48550/arXiv.2402.14905

Lu, Y. (2019). Artificial intelligence: A survey on evolution, models, applications and future trends. Journal of Management Analytics, 6(1), 1–29. https://doi.org/10.1080/23270012.2019.1570365

Lu, Y. (2022). Implementing blockchain in information systems: A review. Enterprise Information Systems, 16(12), 1876–1907. https://doi.org/10.1080/17517575.2021.2008513

Lu, Y. (2023). The blockchain: State-of-the-art and research challenges. Journal of Industrial Information Integration, 15, 80–90. https://doi.org/10.1016/j.jii.2019.04.002

Lu, Y. (2025). The current status and developing trends of Industry 4.0: A review. Information Systems Frontiers, 27(1), 215–234. https://doi.org/10.1007/s10796-021-10221-w

Ma, X., Fang, G., & Wang, X. (2024). The era of 1-bit LLMs: All large language models are in 1.58 bits. arXiv preprint arXiv:2402.17764. https://doi.org/10.48550/arXiv.2402.17764

Malkov, Y. A., & Yashunin, D. A. (2020). Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. IEEE Transactions on Pattern Analysis and Machine Intelligence, 42(4), 824–836. https://doi.org/10.1109/TPAMI.2018.2889473

McInnes, L., Healy, J., & Astels, S. (2017). hdbscan: Hierarchical density based clustering. Journal of Open Source Software, 2(11), 205. https://doi.org/10.21105/joss.00205

Meng, K., Bau, D., Andonian, A., & Belinkov, Y. (2022). Locating and editing factual associations in GPT. In Advances in Neural Information Processing Systems, 35, 17359–17372.

Min, S., Lewis, M., Zettlemoyer, L., & Hajishirzi, H. (2022). MetaICL: Learning to learn in context. In Proceedings of NAACL: Human Language Technologies (pp. 2791–2809). https://doi.org/10.18653/v1/2022.naacl-main.201

Olausson, T. X., Inala, J. P., Wang, C., Gao, J., & Solar-Lezama, A. (2023). Is self-repair a silver bullet for code generation? arXiv preprint arXiv:2306.09896. https://doi.org/10.48550/arXiv.2306.09896

Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., & others. (2022). Training language models to follow instructions with human feedback. In Advances in Neural Information Processing Systems, 35, 27730–27744.

Park, J. S., O'Brien, J., Cai, C. J., Morris, M. R., Liang, P., & Bernstein, M. S. (2023). Generative agents: Interactive simulacra of human behavior. In Proceedings of ACM Symposium on User Interface Software and Technology (pp. 1–22). https://doi.org/10.1145/3586183.3606763

Pei, J., Han, J., Mortazavi-Asl, B., Wang, J., Pinto, H., Chen, Q., Dayal, U., & Hsu, M. C. (2004). Mining sequential patterns by pattern-growth: The PrefixSpan approach. IEEE Transactions on Knowledge and Data Engineering, 16(11), 1424–1440. https://doi.org/10.1109/TKDE.2004.77

Perez, E., Huang, S., Song, F., Cai, T., Ring, R., Aslanides, J., Glaese, A., McAleese, N., & Irving, G. (2022). Red teaming language models with language models. arXiv preprint arXiv:2202.03286. https://doi.org/10.48550/arXiv.2202.03286

Raasveldt, M., & Mühlhäuser, H. (2019). DuckDB: An embeddable analytical database. In Proceedings of ACM SIGMOD International Conference on Management of Data (pp. 1981–1984). https://doi.org/10.1145/3299869.3320212

Ren, R., Agarwal, A., Mazeika, M., Menghini, C., Vacareanu, R., & Kenstler, B. (2025). The MASK benchmark: Disentangling honesty from accuracy in AI systems. arXiv preprint arXiv:2503.03750. https://doi.org/10.48550/arXiv.2503.03750

Russell, S. (2019). Human Compatible: Artificial Intelligence and the Problem of Control. Viking.

Schroff, F., Kalenichenko, D., & Philbin, J. (2015). FaceNet: A unified embedding for face recognition and clustering. In Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (pp. 815–823). https://doi.org/10.1109/CVPR.2015.7298682

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347. https://doi.org/10.48550/arXiv.1707.06347

Shazeer, N., Mirhoseini, A., Maziarz, K., Davis, A., Le, Q., Hinton, G., & Dean, J. (2017). Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In International Conference on Learning Representations.

Solaiman, I., Clark, J., & Brundage, M. (2019). GPT-2: 1.5B release. OpenAI Blog. https://openai.com/blog/gpt-2-1-5b-release

Stiennon, N., Ouyang, L., Wu, J., Ziegler, D. M., Lowe, R., Voss, C., Radford, A., Amodei, D., & Christiano, P. (2020). Learning to summarize from human feedback. In Advances in Neural Information Processing Systems, 33, 3008–3021.

Taori, R., Gulrajani, I., Zhang, T., Dubois, Y., Li, X., Guestrin, C., Liang, P., & Hashimoto, T. B. (2023). Alpaca: A strong, replicable instruction-following model. Stanford Center for Research on Foundation Models. https://crfm.stanford.edu/2023/03/13/alpaca.html

Turpin, M., Michael, J., Perez, E., & Bowman, S. R. (2024). Language models do not always say what they think: Unfaithful explanations in chain-of-thought prompting. In Advances in Neural Information Processing Systems, 36, 74952–74965. https://doi.org/10.48550/arXiv.2211.11876

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. In Advances in Neural Information Processing Systems, 30, 5998–6008.

Wang, J., Yi, X., Guo, R., Jin, H., Xu, P., Li, S., Wang, X., Guo, X., Li, C., Xu, X., & others. (2021). Milvus: A purpose-built vector data management system. In Proceedings of ACM SIGMOD International Conference on Management of Data (pp. 2614–2627). https://doi.org/10.1145/3448016.3457550

Wang, Y., Zhang, H., Li, X., Li, P., Liu, Y., & Zhang, C. (2024). Self-critique improving code generation with large language models. In International Conference on Learning Representations. https://openreview.net/forum?id=wxf6zU0XeR

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Ichter, B., Xia, F., Chi, E., Le, Q., & Zhou, D. (2022). Chain-of-thought prompting elicits reasoning in large language models. In Advances in Neural Information Processing Systems, 35, 24824–24837.

Wu, Y., Pan, X., Hong, G., & Yang, M. (2025). OpenDeception: Benchmarking and investigating AI deceptive behaviors via open-ended simulation. arXiv preprint arXiv:2504.13707. https://doi.org/10.48550/arXiv.2504.13707

Xu, L. D., Lu, Y., & Li, L. (2021). Embedding blockchain technology into IoT for security: A survey. IEEE Internet of Things Journal, 8(13), 10452–10473. https://doi.org/10.1109/JIOT.2021.3060508

Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K., & Cao, Y. (2023). ReAct: Synergizing reasoning and acting in language models. In International Conference on Learning Representations. https://openreview.net/forum?id=WE_vluYUL-X

Yang, L., Hou, Q., Zhu, X., Lu, Y., & Xu, L. D. (2025). Potential of large language models in blockchain-based supply chain finance. Enterprise Information Systems, 19(11), 2541199. https://doi.org/10.1080/17517575.2024.2541199

Yang, X., Wang, X., Zhang, Q., Petzold, L., Wang, W. Y., & Zhao, X. (2023). Shadow alignment: The ease of subverting safely-aligned language models. arXiv preprint arXiv:2310.02949. https://doi.org/10.48550/arXiv.2310.02949

Yuan, Z., Shang, Y., Song, Y., Wu, Q., Yan, Y., Sun, H., Dong, B., Lu, W., Lee, V., Liang, C., & others. (2024). QServe: W4A8KV4 quantization and system co-design for efficient LLM serving. arXiv preprint arXiv:2405.04532. https://doi.org/10.48550/arXiv.2405.04532

Zaki, M. J. (2001). SPADE: An efficient algorithm for mining frequent sequences. Machine Learning, 42(1–2), 31–60. https://doi.org/10.1023/A:1007652502315

Zhang, C., & Lu, Y. (2021). Study on artificial intelligence: The state of the art and future prospects. Journal of Industrial Information Integration, 23, 100224. https://doi.org/10.1016/j.jii.2021.100224

Zhang, H., & Lu, Y. (2025). Web 3.0: Applications, opportunities and challenges in the next internet generation. Systems Research and Behavioral Science, 42(4), 996–1015. https://doi.org/10.1002/sres.3108

Zhu, W., & Bhat, N. (2023). FedPET: Federated parameter-efficient transfer learning for distantly labeled NLP tasks. arXiv preprint arXiv:2012.11560. https://doi.org/10.48550/arXiv.2012.11560

Ziegler, D. M., Stiennon, N., Wu, J., Brown, T. B., Radford, A., Amodei, D., Christiano, P., & Irving, G. (2019). Fine-tuning language models from human preferences. arXiv preprint arXiv:1909.08593. https://doi.org/10.48550/arXiv.1909.08593

Zou, A., Wang, Z., Carlini, N., Nasr, M., Kolter, J. Z., & Fredrikson, M. (2023). Universal and transferable adversarial attacks on aligned language models. arXiv preprint arXiv:2307.15043. https://doi.org/10.48550/arXiv.2307.15043