

ForensicBench: Evaluating Agentic LLMs on Journal-Entry Fraud Detection

Guy Stéphane Waffo Dzuyo^{1,2}, Gaël Guibon^{2,3}, Christophe Cerisara², Luis Belmar-Letelier¹

¹Forvis Mazars

²LORIA, CNRS, Université de Lorraine

³Université Sorbonne Paris Nord, CNRS, Laboratoire d’Informatique de Paris Nord, LIPN

Correspondence: guy.stephane.waffo@forvismazars.com

Abstract

Accounting fraud costs organizations billions annually, yet auditors rarely confront it as a single suspicious posting. Instead, they investigate coordinated *fraud schemes* such as fictitious vendor payments, ghost-employee payroll, and inflated revenues that span many journal entries, each individually routine, and only become visible when understood against the underlying accounting process. Existing AI systems either score isolated entries as anomalies or mine company-level disclosures; none evaluate whether a large language model (LLM) agent can query a live ledger, recover coordinated fraud patterns, and assign the correct scheme type. We release **ForensicBench**: the first benchmark for evaluating agentic LLMs on *scheme-level* journal-entry fraud detection. The dataset, **Forensic Ledger**, extends Data-Synth with five injected fraud scheme types and scheme-level labels linking journal entries to coordinated instances. We evaluate twelve open-weight models under a single reference agent scaffold on a public leaderboard: among them, the best reaches only 34.7% Entry-F1, and Type-F1 is up to 13 points lower. We publicly release the dataset, evaluation harness, and a reference agentic baseline.¹

1 Introduction

Accounting fraud (the deliberate manipulation of financial records to misappropriate assets or misstate results) costs organizations billions annually (Association of Certified Fraud Examiners, 2024). In practice, much of this fraud manifests as *journal-entry (JE) fraud*: coordinated schemes such as fictitious vendor payments, ghost-employee payroll, or revenue inflation that span multiple postings to the *general ledger (GL)*, the canonical chronological record of all financial transactions organised by account class, each posting individually routine. Forensic auditors do not test entries in isolation;

they search for *scheme-level* evidence, linking dispersed postings into a typed pattern (e.g., shadow payroll, vendor collusion) by reasoning over accounting processes such as vendor approval workflows, payroll cycles, and period-end cutoffs. This differs from flagging statistical outliers on single records, and it is the kind of reasoning that agentic LLMs (systems capable of querying databases, executing code, and maintaining investigative state across many steps (Yao et al., 2023)) may perform. We do not claim LLMs outperform classical detectors at entry-level anomaly detection; rather, scheme-level detection is a different, harder task (linking dispersed postings and reasoning over accounting processes), for which entry-level statistical tests provide no scheme-type label. Our uninformed audit-test battery (a Benford first-digit test (Benford, 1938; Nigrini, 2012), a round-amount and a period-end journal-entry test (Lanza, 2006), and an Isolation Forest (Liu et al., 2008)) confirms the gap is behavioral rather than statistical: it reaches only 0.36% Entry-F1 versus 34.7% for the best LLM (Appendix E.2).

No existing benchmark evaluates whether agentic LLMs can discover and type coordinated fraud schemes over a live transactional ledger. Prior work falls short in three respects. *Financial-statement datasets* (Bao et al., 2020; Dechow et al., 2011) label fraud at company-year granularity, far above the scheme level needed for ledger investigation. *Entry-level anomaly detectors* (Hilal et al., 2022; Schreyer et al., 2019) flag isolated statistical outliers but provide no scheme-type labels and no accounting-process context; they cannot evaluate whether a system understands *which* coordinated pattern a set of entries constitutes. *General agent benchmarks* (Liu et al., 2023; Mialon et al., 2023) cover web browsing and code tasks but include no forensic accounting setting. What is needed is a realistic multi-sector ledger with scheme-level ground truth; ForensicBench uses the French *Plan*

¹https://github.com/WaguyMz/Forensic_Bench.

Comptable Général (PCG) (Autorité des Normes Comptables, 2014), a hierarchical chart of accounts whose numbered classes encode balance-sheet and income-statement structure and require agents to navigate a realistic account tree.

We introduce **ForensicBench** to close this gap: a reproducible benchmark for scheme-level journal-entry fraud detection that evaluates whether agentic systems can discover and type coordinated fraud patterns over realistic enterprise ledgers, instead of scoring isolated postings. We summarize our contributions as follows: (1) **Forensic Ledger**, a realistic synthetic corpus across five industry sectors with five injected fraud scheme types and document-level ground truth at both journal-entry and scheme-type granularity; (2) **ForensicBench Evaluation Protocol**, scoring Entry-F1, Type-F1, recall, precision, scheme coverage, and cross-seed consistency; (3) **Reference Agent**, an agentic baseline that investigates ledgers via SQL and code execution; and (4) **Open Forensic Leaderboard**, a public ranking of open-weight models from large to small on the benchmark.

2 Related Work

Entry-level anomaly detection. Computational audit research has predominantly addressed individual-entry testing: rule-based journal-entry tests (JETs; Benford tests, round-amount filters), multivariate outlier detectors, deep autoencoders over enterprise resource planning (ERP) records (Hilal et al., 2022; Schreyer et al., 2017, 2019), and graph neural network (GNN) methods that exploit the relational structure of double-entry book-keeping (Sotiropoulos et al., 2023; Huang et al., 2026). AuditCopilot extends this line by prompt-tuning LLMs on JSON-serialised postings grouped by posting ID (a single balanced debit-credit pair) with Isolation Forest hints (Kadir et al., 2025); its unit of analysis remains a single balanced transaction and it does not link dispersed postings into labelled coordinated schemes or evaluate scheme discovery over a full relational ledger.

Agentic investigation and benchmarks. AuditAgent addresses a different problem: cross-document evidence-chain localisation over unstructured corporate disclosures (annual reports, regulatory filings) using variational Bayesian subject priors and multi-expert consensus (Bai et al., 2025). Its input modality (free-text financial statements) and evaluation target (evidence spans in narrative

documents) differ from ForensicBench’s structured transactional ledgers with entry-resolution scheme-level ground truth. General agent benchmarks (AgentBench, GAIA (Liu et al., 2023; Mialon et al., 2023)) assess web and code tasks; ReAct (Yao et al., 2023) and Toolformer (Schick et al., 2023) provide the interleaved reasoning-tool paradigm our reference agent builds on. None supply live relational accounting ledgers, coordinated scheme-level ground truth, or forensic evaluation criteria.

Synthetic accounting data. Privacy constraints motivate synthetic ledgers; prior generators produce records respecting double-entry constraints (Motai et al., 2025; Schultz and Tropmann-Frick, 2020; Kröger and Schultz, 2025), and DataSynth provides realistic chart-of-accounts structure with master data (Ivertowski, 2026). No prior synthetic resource combines controlled injection of process-linked, multi-entry fraud schemes with *scheme-level* ground truth linking each entry to a labelled coordinated fraud instance, which is needed to evaluate scheme discovery rather than entry-level anomaly scoring.

3 Forensic Ledger Dataset

The Forensic Ledger is built in two stages: we generate realistic enterprise ledgers with an open-source synthetic data engine (§3.1), then inject coordinated, multi-entry fraud schemes with document-level ground truth (§3.2). We instantiate this pipeline across five industry sectors (§3.3) and expose the result to agents through a read-only relational interface with a hidden label store (§3.3).

3.1 Base Ledger Generator

We build on DataSynth (Ivertowski, 2026), an open-source enterprise data generator that produces interconnected financial records for a configurable firm: general-ledger (GL) journal entries, procure-to-pay and order-to-cash document flows (purchase orders, goods receipts, invoices, payments), sub-ledgers, and master data for vendors, employees, customers, materials, and fixed assets. Generation is governed by accounting structure rather than free-form sampling (Motai et al., 2025; Kröger and Schultz, 2025): every journal entry is balanced at construction (debits equal credits), amounts follow log-normal distributions calibrated for first-digit Benford compliance (Benford, 1938; Nigrini, 2012), postings respect business-day and period-end calendars, and a deterministic seed makes each

ledger reproducible. All entries are denominated in EUR under the French *Plan Comptable Général* (PCG)², so that account semantics are consistent across sectors.

Out of the box, the generator targets *entry-level* irregularities (statistical outliers, duplicate or reversed postings, wrong-period or wrong-account errors, and isolated fraud markers), each labelled at the level of a single record. These labels support conventional anomaly detection but describe individual postings in isolation: they do not encode the coordinated, multi-entry structure that characterises real fraud schemes. The public v0.6.2 release ships a nascent multi-stage scheme module with two implementations only (*gradual embezzlement*, *vendor kickback*); we extend this into five canonical benchmark types, add three further scheme families, and supply multi-perpetrator co-occurrence, sector calibration, and dual-granularity labels for evaluation.³ We build on the generator’s realistic ledger backbone and add a labelling and scheme-injection layer for ForensicBench evaluation.

3.2 Injected Fraud Schemes

A *fraud scheme* is a coordinated sequence of journal entries, posted over days to months, that jointly enacts a fraudulent objective while each entry remains individually plausible. We extend the generator’s two-scheme module into a full scheme-injection engine and model schemes as multi-stage processes: once per simulated month, each scheme type may start under an independent Bernoulli draw; active schemes then advance day by day, emitting stage-dependent journal entries (e.g., a fictitious invoice followed weeks later by its matching payment). Algorithm 1 summarises the full pipeline; scheme-specific stage logic is implemented per type in the release. Our extension adds the forensic structure observed in practice: a fixed perpetrator (*created_by*) spanning incompatible roles, after-hours timestamps, reference reuse, period-end clustering, ghost-employee master-data mutation, and concealment postings through sus-

pense accounts, together with perpetrator reuse and co-occurrence so that a single perpetrator may operate multiple linked schemes.

Scheme lineage. ForensicBench injects five canonical scheme types (Appendix 11) spanning principal occupational-fraud families (Association of Certified Fraud Examiners, 2024). Two inherit from DataSynth v0.6.2 (*gradual embezzlement*, *vendor kickback*), canonicalised as **fictitious account payables disbursements** (salami-slicing and expense-laundering variants over procure-to-pay) and **vendor collusion** (kickback and related-party variants). We add three families absent from the public module: **revenue manipulation** (order-to-cash period-end window dressing), **inventory manipulation** (goods receipts without purchase orders (PO), and year-end write-offs), and **shadow payroll** (ghost-employee payroll accruals), extending coverage to order-to-cash, supply-chain, and hire-to-retire processes beyond procure-to-pay. Variant dispatch within each type preserves a single canonical label.

Document-level ground truth. Each scheme instance is assigned a persistent identifier, and every journal entry it emits receives a unique action identifier that joins one-to-one to a label record. A label thus links a specific *document_id* to (i) its *scheme type* and (ii) the *scheme instance* that groups all entries of the same coordinated fraud. This dual granularity is what enables scheme-level evaluation: we can score both whether an agent flags the right entries and whether it recovers and types the coordinated pattern they form, rather than only whether it detects isolated anomalies.

3.3 Sector Configurations and Agent Interface

We instantiate the pipeline across five sectors (Energy, Healthcare, Luxury Goods, Manufacturing, Transport), each a distinct firm sharing the same currency (EUR), accounting framework (French PCG), and horizon (~100K journal entries over 2023–2025), but with its own chart of accounts, process mix, and per-sector scheme calibration (e.g. shadow payroll is prominent in Healthcare and Transport; inventory manipulation in Manufacturing and Luxury Goods). Agents access each ledger as a read-only PostgreSQL database; ground-truth labels reside in a separate store never visible to the agent. The schema documents an *hr_employees* table as the primary tool for shadow-payroll investigation, yet the table is access-controlled at

²The PCG is France’s mandatory general ledger chart of accounts, codified by the *Autorité des Normes Comptables* in Regulation ANC n°2014-03 (*Autorité des Normes Comptables*, 2014).

³DataSynth v0.6.2 implements *embezzlement.rs* and *kickback.rs*; we canonicalise these as fictitious accounts payable (AP) disbursements and vendor collusion and add revenue manipulation, inventory manipulation, and shadow payroll (Appendix 11).

runtime (permission denied); an effective investigator must treat this as a dead end and pivot to a GL-structural path, testing recovery after denied database access (see Appendix D).

4 ForensicBench Evaluation

ForensicBench evaluates agents on two levels: *entry-level* detection quality (which journal entries are flagged, and how correctly) and *scheme-level* coverage (how much of each coordinated fraud pattern an agent has uncovered). Six axes together define the benchmark; no single aggregate score is used, so each metric remains readable on its own.

4.1 Task Formulation

An agent receives read-only access to a sector ledger containing journal-entry headers and lines, chart of accounts, and vendor, employee, and customer master data. No fraud labels, anomaly scores, or scheme identifiers are visible; the agent must derive all evidence from the transactional data alone. The agent reports its findings by flagging individual journal entries and assigning each a scheme type drawn from the five canonical labels (Appendix 11). Evaluation matches the reported identifiers against the hidden ground-truth store.

Token-budget constraint. ForensicBench is a *token-budget constrained* benchmark: each run is allocated a fixed global budget of 20M tokens (prompt + reasoning), and the agent is halted when this budget is exhausted. This constraint is a first-class benchmark property, not an implementation detail; it reflects the cost and latency limits of real audit engagements and forces agents to prioritise their investigation strategy rather than exhaustively querying the database.

4.2 Metrics

Let $\mathcal{F}$ be the set of ground-truth fraudulent journal entries and $\hat{\mathcal{F}}$ the set reported by the agent, with $n_c = |\hat{\mathcal{F}} \cap \mathcal{F}|$.

Entry-F1. The primary leaderboard axis.

$$P = \frac{n_c}{|\hat{\mathcal{F}}|}, \quad R = \frac{n_c}{|\mathcal{F}|}, \quad \text{Entry-F1} = \frac{2PR}{P + R}.$$

Recall and Precision are reported separately as independent axes to surface different failure modes: a low-recall/high-precision agent is overly conservative; a high-recall/low-precision agent flags indiscriminately.

Type F1. A detection is a *typed true positive* if its `document_id` $\in \mathcal{F}$ and its scheme type matches the ground-truth label; Type-F1 uses the same denominators as Entry-F1 with this stricter count, ensuring $\text{Type-F1} \leq \text{Entry-F1}$.

Coverage. Entry-F1 measures entry-level accuracy, but an agent that flags scattered entries from many schemes without ever reconstructing a full coordinated pattern scores the same as one that recovers complete schemes. Coverage addresses this: for each ground-truth scheme instance s with document set $\mathcal{D}_s$, $\text{cov}(s) = |\hat{\mathcal{F}} \cap \mathcal{D}_s| / |\mathcal{D}_s|$, and leaderboard Coverage is the macro-mean over *all* instances (including zero-coverage ones), rewarding breadth of investigation rather than concentration on easy entries.

Consistency. A single run at temperature zero is insufficient to characterise a model: tool-call ordering and provider-side sampling introduce variance that can swing Entry-F1 by tens of points within the same model (see Appendix B). Consistency $= \max(0, 1 - \sigma/(\mu + \varepsilon)) \times 100$ quantifies this, where $\{f_i\}_{i=1}^5$ are per-replicate Entry-F1 values; it ranks last in the lexicographic ordering because a stably-wrong model scores high.

4.3 Protocol and Ranking

Each model runs on all five sectors with five independent replicates per sector, yielding 25 evaluation runs; a model enters the leaderboard only if all 25 runs are complete, preventing cherry-picking of favourable sectors. Per-run metrics are averaged over the five replicates within each sector, then macro-averaged (unweighted) across sectors. Unweighted averaging does not correct for scheme-population imbalance (~ 394 fictitious account payables vs. ~ 56 shadow payroll entries per run); per-scheme recall in Appendix B provides the imbalance-aware view. Models are ranked by Entry-F1, then Type-F1, Recall, Precision, Coverage, and Consistency, rather than by a composite score, so that performance on each axis remains independently interpretable.

5 Reference Agentic System

ForensicBench is model-agnostic: any agent that can query a SQL database and call `report_suspicion` is a valid participant. For the initial leaderboard, we use a reference system with a *plan-then-investigate* workflow, drawing on

tool-augmented LLMs (Schick et al., 2023), plan-before-execute reasoning (Wang et al., 2023), and multi-agent parallelism (Wu et al., 2024), adapted to the forensic setting. The reference system is intentionally simple: it avoids reflection, replanning, and inter-worker negotiation so that results are straightforward to reproduce. Stronger agents (e.g., with replanning or multi-agent coordination) are expected to score higher; we release the harness to support such submissions.

5.1 Orientation

Before any fraud-specific reasoning, the agent spends a fixed fraction of its token budget on a *ledger orientation* phase, inspired by the preliminary analytics step in expert financial investigations. The agent issues SQL queries to profile the dataset: it surveys the chart of accounts, estimates transaction volumes per account class, identifies the dominant business processes, and builds an entity-level map of vendors, employees, and customers. This orientation digest (chart-of-accounts structure, amount distributions, posting patterns) is injected as shared context into all subsequent reasoning steps, grounding hypotheses in the actual ledger rather than in generic accounting priors. The orientation phase does not emit any detections; it is strictly investigative setup.

5.2 Hypothesis Planning

After orientation, the orchestrator issues a single planning call (Wang et al., 2023) that produces a structured investigation plan: for each of the five scheme types, it formulates a ranked set of falsifiable hypotheses (e.g., “payments to vendors registered within 30 days of the transaction have no matching purchase order”), each assigned an explicit dispatch priority, a set of exit criteria, and a token budget drawn from the remaining global allowance. Hypotheses with higher priority are dispatched first, allowing the agent to prioritise the highest-priority hypotheses when the budget is tight. The plan is fixed after this one-shot step; the hypothesis backlog is not modified during execution.

5.3 Sequential Hypothesis Workers

Each hypothesis is executed by an independent *worker agent* in a bounded ReAct loop (Yao et al., 2023): the worker alternates between reasoning steps and tool calls until it satisfies its exit criteria or exhausts its token allocation. Workers have

access to **SQL** (*preview* and *export* over journal-entry headers and lines, chart of accounts, and master data), a Python **code interpreter** (Gao et al., 2023) (pandas, NumPy, SciPy, scikit-learn, NetworkX; Benford tests, time-clustering, graph analysis, fuzzy matching), a **scratchpad**, **grep**, **read_image**, and **report_suspicion**. Workers also write to a shared *blackboard* through which salient entity-level findings (e.g., a vendor sharing a bank account with an employee) are made visible to workers investigating related schemes, without exposing full message histories. For the leaderboard results reported here, workers execute sequentially by dispatch priority to encourage reproducibility; the architecture supports parallel execution (Wu et al., 2024) and we release both configurations. Detections are merged programmatically (no LLM reconciliation pass), and the union of reported entries forms the agent’s final submission.

5.4 Prompt Design and Label Isolation

The *fraud catalogue* (Appendix F) is a **standardised benchmark artefact** released with the evaluation harness: it describes each of the five scheme types in terms of normal process context and conceptual fraud indicators, and every leaderboard participant must inject it verbatim into their agent’s system prompt. This design maximises reproducibility across agent submissions while ensuring no information from the injection process leaks to the agent: the catalogue explicitly withholds all GL account codes, posting templates, amount parameter ranges, and calendar cutoffs used during data generation. All concrete indicators (GL accounts, SQL filters, amount thresholds, timing windows) must be derived by the agent from its own ledger profiling, not read off from the catalogue. Full prompt templates are provided in Appendix G.

6 Experiments

Experimental setup. All 300 leaderboard runs use the reference agent (§5) at temperature zero under the 20M-token budget (§4), serving open-weight checkpoints via vLLM (Kwon et al., 2023) on four NVIDIA H200 GPUs (147GB RAM each).

6.1 Models and Results

We evaluate 12 open-weight models from seven providers (Table 1), spanning 4B to 397B parameters, with no task-specific fine-tuning. Macro-averaged ForensicBench scores are reported over

Model	Provider	Params	E-F1	Type-F1	Recall	Prec.	Cover.	Consist.
MiniMax-M2.7 ^a	MiniMax	230B	34.7	21.7	29.0	48.4	21.7	50.7
Qwen3.5-397B ^b	Alibaba	397B	26.3	14.1	20.5	41.3	18.0	61.4
Qwen3.5-122B ^b	Alibaba	122B	25.2	13.5	19.8	40.6	14.2	46.9
Qwen3.6-35B ^b	Alibaba	35B	15.0	9.4	9.8	36.6	9.3	28.1
Mistral-Medium-3.5-128B ^c	Mistral AI	128B	13.6	9.1	10.3	21.8	9.5	43.8
Gemma-4-31B ^d	Google AI	31B	12.1	8.7	7.5	37.4	6.8	76.1
Gemma-4-E4B ^d	Google AI	4B	8.5	3.8	5.2	29.5	4.2	31.2
GPT-OSS-120B ^e	OpenAI	120B	8.0	5.2	4.6	39.9	4.4	55.8
Mistral-Small-4-119B ^c	Mistral AI	119B	3.8	2.4	2.3	13.4	2.1	28.5
Qwen3.5-9B ^b	Alibaba	9B	3.5	2.2	2.0	18.5	2.0	18.0
Granite-30B ^f	IBM	30B	2.8	1.5	1.5	16.9	1.6	38.4
Llama-3.3-70B ^g	Meta	70B	0.3	0.0	0.1	3.2	0.2	80.4

Table 1: ForensicBench leaderboard (macro-average over 25 runs per model, 300 runs total), ranked by Entry-F1. Bold indicates the best score in each column; E-F1 = Entry-F1; Prec. = Precision; Cover. = Coverage; Consist. = Consistency (0–100); all other metrics are percentages. The current ranking is determined by Entry-F1 alone (no ties occurred); the lexicographic tie-breaker is future-proofing. Bootstrap 95% confidence intervals over the 25 runs are reported in Appendix E.1. References: ^aMiniMax Team (2026); ^bQwen Team (2026); ^cMistral AI (2026a), Mistral AI (2026b) (Mistral AI); ^dGemma Team, Google AI (2026); ^eOpenAI (2025); ^fGranite Team, IBM (2026); ^gMeta (2024).

25 runs per model (5 sectors $\times$ 5 replicates). Overall performance is low: even the best evaluated model reaches only 34.7% Entry-F1 under the reference scaffold, confirming that scheme-level forensic reasoning over transactional ledgers remains difficult for current open-weight LLMs. We report bootstrap 95% confidence intervals over the 25 runs per model (Appendix E.1); for 8 of 11 adjacent leaderboard entries the confidence intervals overlap, and only 4 of 11 pairwise differences are significant, so adjacent mid-tier rankings should not be read as statistically distinguishable at $n = 5$ replicates per sector. Scheme-level classification is harder still: Type-F1 trails Entry-F1 by up to 13 points on every model (range 0.3–13.0), meaning agents that detect fraud entries often misidentify the coordinated pattern they belong to.

A hand-engineered rule-based oracle with full database access achieves 78–91% recall on the four rule-amenable schemes and 83–86% on shadow payroll, confirming that every injected scheme is recoverable from the ledger (Appendix D). Low LLM recall therefore reflects model capability, not label noise or undetectable injections.

6.2 Discussion

Scheme difficulty and what drives the ranking. Revenue manipulation is the most accessible scheme: 11 of 12 models attain non-zero recall on it, and it is detected in 48–100% of runs (Table 6). The leaderboard is instead set by **fictitious account payables disbursements** and **shadow**

payroll: only MiniMax-M2.7 exceeds recall on revenue manipulation alone (13.1% on the former; 1.6% on the latter, seven models at zero). The difficulty gap reflects process depth: revenue manipulation concentrates in a single account at period-end, while fictitious account payables (AP) and shadow payroll require cross-table reasoning across weeks of interleaved postings spanning multiple accounting sub-processes. Parameter count does not predict Entry-F1: Gemma-4-E4B (4B) outperforms GPT-OSS-120B (120B), and MiniMax-M2.7 and the Qwen3.5 family rank highest while also leading on AIME 2026 and GPQA-Diamond (MiniMax Team, 2026; Qwen Team, 2026).

Cross-sector variance. Sector difficulty varies substantially (macro Entry-F1: 9.8–17.9%; Appendix B), and high Consistency can reflect stable failure modes rather than robust success (e.g., Llama-3.3-70B at 80.4 Consist. with 0.3% Entry-F1); the five-replicate design is therefore necessary for reliable rankings.

Failure modes. Audit traces (Appendix C) reveal recurring errors: (i) high token use, low yield (Qwen); (ii) early termination (GPT-OSS-120B, Gemma-4-31B, Granite-30B); (iii) over-flagging without verification (Mistral models); (iv) report_suspicion misuse (Llama-3.3-70B, Gemma-4-E4B); and weak scheme typing (Mistral-Medium-3.5-128B, Llama-3.3-70B). On shadow payroll, seven models score 0% recall after permission denied, and under the reference scaffold

fold (which fixes its investigation plan after one call and cannot replan) none of the evaluated models pivot to general-ledger analysis (Appendix D).

Single-scaffold measurement. Because all twelve models are evaluated through the same reference scaffold, the leaderboard measures model $\times$ scaffold rather than model capability in isolation; a stronger scaffold (replanning, multi-agent coordination) or a frontier model is expected to shift absolute scores, and the harness is released for the community to submit such variants.

7 Conclusion

We introduced **ForensicBench**, the first benchmark to evaluate whether agentic LLMs can discover coordinated fraud schemes over live relational ledgers. The benchmark pairs the **Forensic Ledger** (five sectors, five scheme types, entry- and scheme-level ground truth) with a six-axis protocol, a reference agent, and an open leaderboard. Across 300 runs over 12 open-weight models under the reference scaffold, performance remains far from audit-ready: the best evaluated system (MiniMax-M2.7) reaches only 34.7% Entry-F1, leaving more than two-thirds of fraudulent entries undetected. The hardest cases are cross-table schemes such as **fictitious account payables disbursements** and **shadow payroll**, which require tracing coordinated patterns across weeks of interleaved postings.

We publicly release the benchmark and welcome the community to submit new models, propose novel agentic strategies, and extend the dataset with more complex, multi-scheme scenarios.

Acknowledgments

Experiments presented in this paper were carried out using both the Grid’5000 testbed and the inIdEx projet ressources. Grid’5000 is supported by a scientific interest group hosted by Inria and including CNRS, RENATER and several Universities as well as other organizations (see <https://www.grid5000.fr>). This work has received support from the investment programme "France 2030" as part of the IdEx programme (ANR-18-IDEX-0001) implemented by Université Paris Cité, under which the inIdEx project EFL is conducted. Especially, the inIdEx project provided the NVIDIA H200 GPUs, which together with Grid’5000 made the 300 forensic investigations (12 models $\times$ 5 sectors $\times$ 5 seeds) feasible.

We are grateful to the authors of DataSynth (Ivertowski, 2026), whose open-source enterprise data generator opened the way to systematic research in a field long hampered by the scarcity of proprietary, real-world ledgers.

Limitations

Leaderboard coverage. The current leaderboard is restricted to open-weight models for which we could run the full 5×5 evaluation grid within our compute budget. Several frontier open-weight systems are therefore absent, most notably Kimi K2 (1T parameters) (Kimi Team, 2025) and other recently released mixture-of-experts flagships whose API or inference cost made 300 full forensic runs (~ 20 M tokens each) infeasible at the time of submission. Closed-source proprietary models (GPT-4o, Claude 3.5 Sonnet, Gemini 1.5 Pro) are similarly excluded; their absence means ForensicBench has not yet been stress-tested against the strongest available systems. We welcome community submissions to expand the leaderboard and will maintain it as a living benchmark.

Leaderboard governance. To limit leaderboard gaming, submissions must disclose their agent design; we audit the submission trail to verify that a submission does not hard-code the injection rules that are visible in the released generator code; and we will expand the dataset with new scheme variants over time so that rule memorisation does not transfer.

Synthetic-only evaluation. ForensicBench is evaluated exclusively on synthetic data. While the Forensic Ledger is calibrated for Benford compliance, double-entry balancing, and process realism, synthetic generators operate on simplified statistical manifolds and cannot reproduce the long-tail irregularities, enterprise resource planning (ERP) migration artefacts, period-end manual corrections, and messy master-data inconsistencies found in authentic corporate ledgers. A model ranking highly on ForensicBench demonstrates strong reasoning over a well-structured synthetic environment; whether these capabilities transfer to production audit settings requires validation on real anonymised ledgers, which we identify as important future work.

Non-determinism. Even at temperature zero, the reference agent does not guarantee fully deterministic runs: tool-call ordering, network latency,

and provider-side sampling can introduce variance across seeds, which is why we report five replicates per sector. The 5×5 protocol mitigates but does not eliminate this variance, and the Consistency metric captures its residual effect. More deterministic evaluation approaches (e.g., constrained decoding or fixed execution traces) would strengthen reproducibility guarantees and are an open direction for future work.

Ethical Considerations

Assistive role. ForensicBench is designed to evaluate agents as decision-support tools for forensic auditors, not as autonomous systems that make findings of misconduct. In any realistic deployment, flagged entries are hypotheses for a qualified professional to review and corroborate; responsibility for any conclusion rests with the accountable human auditor, not the model. We do not expect agentic systems to replace human auditors, but to help them prioritise and cover more ground within the time and cost constraints of real engagements.

False-positive harm. A system that incorrectly flags journal entries can damage the reputation of employees or vendors named in a false positive. This makes human oversight non-negotiable in any production use, and it is a key reason the benchmark reports Precision as an independent axis rather than optimising for recall alone.

Dual-use. Publishing fraud-detection benchmarks could, in principle, help bad actors understand which patterns are detectable and design harder-to-detect schemes. We mitigate this risk by describing schemes at the level of accounting process concepts rather than releasing specific GL account codes, timing parameters, or injection thresholds; the fraud catalogue operates at the same level of abstraction that any trained auditor already knows.

Privacy. The Forensic Ledger contains entirely synthetic records generated from statistical distributions; no real individuals, companies, or transactions are present. No personally identifiable or commercially sensitive data is released.

References

Association of Certified Fraud Examiners. 2024. Occupational fraud 2024: A report to the nations. <https://www.acfe.com/rttt>.

Autorité des Normes Comptables. 2014. [Règlement relatif au plan comptable général \(anc n°2014-03\)](#).

Songran Bai, Bingzhe Wu, Yiwei Zhang, Chengke Wu, Xiaolong Zheng, Yaze Yuan, Ke Wu, and Jianqiang Li. 2025. [AuditAgent: Expert-guided multi-agent reasoning for cross-document fraudulent evidence discovery](#). In *Proceedings of the Sixth ACM International Conference on AI in Finance*.

Yang Bao, Bin Ke, Bin Li, Y Julia Yu, and Jie Zhang. 2020. Detecting accounting fraud in publicly traded US firms using a machine learning approach. *Journal of Accounting Research*, 58(1):199–235.

Frank Benford. 1938. The law of anomalous numbers. *Proceedings of the American Philosophical Society*, 78(4):551–572.

Patricia M Dechow, Weili Ge, Chad R Larson, and Richard G Sloan. 2011. Predicting material accounting misstatements. *Contemporary Accounting Research*, 28(1):17–82.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023. PAL: Program-aided language models. In *International Conference on Machine Learning*, pages 10764–10799. PMLR.

Gemma Team, Google AI. 2026. [Gemma 4 model card](#).

Granite Team, IBM. 2026. [Granite 4.1 LLMs: How they’re built](#).

Waleed Hilal, S Andrew Gadsden, and John Yawney. 2022. Financial fraud: A review of anomaly detection techniques and recent advances. *Expert Systems with Applications*, 193:116429.

Qing Huang, Marco Schreyer, Nilson Michiles, and Miklos A. Vasarhelyi. 2026. [Connecting the dots: Graph neural networks for auditing accounting journal entries](#). *Auditing: A Journal of Practice & Theory*.

Michael Ivertowski. 2026. [DataSynth: Synthetic enterprise data generation for ML training, audit analytics, and system testing](#).

Md Abdul Kadir, Sai Suresh Macharla Vasu, Sidharth S. Nair, and Daniel Sonntag. 2025. AuditCopilot: Leveraging LLMs for fraud detection in double-entry bookkeeping. *arXiv preprint arXiv:2512.02726*.

Kimi Team. 2025. [Kimi K2: Open agentic intelligence](#). *Preprint*, arXiv:2507.20534.

Tobias Kröger and Martin Schultz. 2025. Generation of synthetic accounting data in audit: A comparison of tabular generation methods. In *AMCIS 2025 Proceedings*.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient

- memory management for large language model serving with PagedAttention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, pages 611–626.
- Richard B. Lanza. 2006. [Proactive journal entry testing: Detecting entries made in the middle of the night](#). *Fraud Magazine (Association of Certified Fraud Examiners)*.
- Fei Liu, Kai-Wei Ting, and Zhi-Hua Zhou. 2008. Isolation Forest. In *2008 Eighth IEEE International Conference on Data Mining (ICDM 2008)*, pages 413–422. IEEE.
- Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, and 1 others. 2023. Agent-Bench: Evaluating LLMs as agents. *arXiv preprint arXiv:2308.03688*.
- Meta. 2024. [Llama 3.3 70b instruct model card](#).
- Grégoire Mialon, Clémentine Fourier, Craig Swift, Thomas Wolf, Yann LeCun, and Thomas Scialom. 2023. GAIA: A benchmark for general AI assistants. *arXiv preprint arXiv:2311.12983*.
- MiniMax Team. 2026. [The MiniMax-M2 series: Mini activations unleashing max real-world intelligence](#). *Preprint*, arXiv:2605.26494.
- Mistral AI. 2026a. [Mistral medium 3.5](#).
- Mistral AI. 2026b. [Mistral small 4 \(119B\)](#).
- Ryoki Motai, Sota Mashiko, Yuji Kawamata, Ryota Shin, and Yukihiro Okada. 2025. Generating synthetic journal-entry data using variational autoencoder. *Intelligent Systems in Accounting, Finance and Management*, 32(1).
- Mark J. Nigrini. 2012. *Benford’s Law: Applications for Forensic Accounting, Auditing, and Fraud Detection*. John Wiley & Sons, Hoboken, NJ.
- OpenAI. 2025. [gpt-oss-120b & gpt-oss-20b model card](#). *Preprint*, arXiv:2508.10925.
- Qwen Team. 2026. [Qwen3.5-397B-A17B model card](#).
- Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. In *Advances in Neural Information Processing Systems*.
- Marco Schreyer, Timur Sattarov, Damian Borth, Andreas Dengel, and Bernd Reimer. 2017. Detection of anomalies in large-scale accounting data using deep autoencoder networks. *arXiv preprint arXiv:1709.05254*.
- Marco Schreyer, Timur Sattarov, Christian Schulze, Bernd Reimer, and Damian Borth. 2019. Detection of accounting anomalies in the latent space using adversarial autoencoder neural networks. In *Proceedings of the 2nd KDD Workshop on Anomaly Detection in Finance (KDD-Finance ’19)*. ArXiv:1908.00734.
- Martin Schultz and Marina Tropmann-Frick. 2020. [Autoencoder neural networks versus external auditors: Detecting unusual journal entries in financial statement audits](#). In *Proceedings of the 53rd Hawaii International Conference on System Sciences*, pages 5421–5430.
- Konstantinos Sotiropoulos, Lingxiao Zhao, Pierre Jinghong Liang, and Leman Akoglu. 2023. ADAMM: Anomaly detection of attributed multi-graphs with metadata: A unified neural network approach. *arXiv preprint arXiv:2311.07355*.
- Lei Wang, Wanyu Xu, Yihuai Lan, Zhiqiang Hu, Yunshi Lan, Roy Ka-Wei Lee, and Ee-Peng Lim. 2023. Plan-and-solve prompting: Improving zero-shot chain-of-thought reasoning by large language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics*, pages 2609–2634.
- Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Shaokun Zhang, Erkang Zhu, Beibin Li, Li Jiang, Xiaoyun Zhang, and Chi Wang. 2024. [AutoGen: Enabling next-gen LLM applications via multi-agent conversation](#). In *Conference on Language Modeling (COLM)*.
- Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations*.

A Fraud Scheme Injection Algorithm

Forensic Ledger generation proceeds in two phases: (i) produce a realistic, fraud-free enterprise ledger with DataSynth, then (ii) inject coordinated multi-entry schemes and write scheme-level labels. The injection engine extends DataSynth’s two public scheme implementations (*gradual embezzlement*, *vendor kickback*) with three additional families (*revenue manipulation*, *inventory manipulation*, *shadow payroll*), canonicalised as five benchmark types (Appendix 11). It is deterministic given a seed and sector configuration θ (start probabilities, concurrency caps, and process mix). Each emitted journal entry receives a persistent document_id; every scheme action receives an action_id linked to a scheme_instance_id and canonical scheme_type in the hidden label store (see §3.2).

Model	En	HI	Lx	Mn	Tr
MiniMax-M2.7	24.3±12.4	42.6±16.5	32.5±26.1	23.6±11.4	50.4±18.0
Qwen3.5-397B	16.3±8.6	15.9±5.1	35.0±18.4	25.1±9.7	39.3±10.1
Qwen3.5-122B	23.7±10.4	30.1±27.0	20.5±14.5	12.0±5.5	39.7±11.1
Qwen3.6-35B	8.4±8.4	15.3±11.5	12.8±13.4	20.8±12.7	17.8±8.4
Mistral-Medium-3.5-128B	12.3±4.1	10.2±6.4	20.9±9.5	7.3±8.6	17.2±8.9
Gemma-4-31B	8.8±3.0	5.6±1.2	20.9±5.2	11.3±2.8	14.0±3.8
Gemma-4-E4B	9.7±7.5	6.4±2.9	8.0±3.1	9.0±12.2	9.6±10.5
GPT-OSS-120B	8.4±4.8	5.2±1.8	11.8±6.1	7.6±4.1	6.8±3.5
Mistral-Small-4-119B	3.0±2.3	1.6±1.4	4.3±4.0	2.9±4.3	7.3±5.5
Qwen3.5-9B	1.4±1.5	4.3±5.3	2.2±3.8	4.1±4.5	5.8±4.2
Granite-30B	1.1±1.4	3.3±2.1	1.7±2.6	1.5±1.2	6.3±3.3
Llama-3.3-70B	0.0±0.0	0.7±0.9	0.0±0.0	0.0±0.0	0.7±0.8

Table 2: Entry-F1 (%) by sector.

Model	En	HI	Lx	Mn	Tr
MiniMax-M2.7	39.3±9.9	61.2±15.9	41.6±36.1	40.9±21.9	59.2±14.1
Qwen3.5-397B	31.8±14.6	36.8±7.8	43.1±21.7	42.6±15.5	52.4±12.9
Qwen3.5-122B	34.9±13.7	45.6±24.5	27.0±22.2	36.0±13.0	59.7±16.9
Qwen3.6-35B	27.0±25.1	33.5±13.8	25.4±21.0	53.6±22.6	43.3±14.6
Mistral-Medium-3.5-128B	22.9±7.9	22.2±11.1	24.3±13.7	13.3±13.9	26.3±10.8
Gemma-4-31B	33.4±7.2	24.0±5.2	45.3±6.8	46.3±14.9	38.1±13.0
Gemma-4-E4B	31.6±14.2	33.8±15.3	31.2±27.0	27.7±23.0	23.2±13.3
GPT-OSS-120B	47.8±11.8	43.8±14.3	36.2±12.9	44.8±23.9	26.6±11.5
Mistral-Small-4-119B	11.7±9.7	9.3±9.8	13.2±9.7	12.4±15.3	20.1±11.4
Qwen3.5-9B	8.0±8.9	27.5±26.6	8.9±15.7	20.8±17.7	27.5±21.4
Granite-30B	5.2±8.2	29.7±15.6	9.6±12.5	10.6±11.9	29.2±15.0
Llama-3.3-70B	0.0±0.0	11.6±10.2	0.0±0.0	0.0±0.0	4.4±5.0

Table 3: Entry Precision (%) by sector.

Each scheme type implements ADVANCE as a finite-stage state machine: stages specify posting templates, amount ranges, inter-stage delays, and concealment techniques (e.g., matched invoice–payment pairs for fictitious AP, month-end payroll accruals without social-charge lines for shadow payroll). Variant dispatch within a type preserves a single canonical label so evaluation scores scheme family, not implementation detail. Per-sector calibration adjusts start probabilities $p_\tau(\theta)$ and dominant process exposure (Appendix 11) while holding currency, PCG framework, and simulation horizon fixed across sectors.

B Detailed Per-Dataset Performance

B.1 Entry-F1, Precision, and Recall by Sector

Tables 2–4 report mean $\pm$ standard deviation over five independent random seeds per sector (25 runs per model). Sector abbreviations: En = Energy, HI = Healthcare, Lx = Luxury Goods, Mn = Manufacturing, Tr = Transport.

B.2 Sector Difficulty and Run-to-Run Variance

Mean Entry-F1 pooled across all 12 models: Transport 17.9% (easiest), Luxury Goods 14.2%, Healthcare 11.8%, Manufacturing 10.4%, Energy 9.8% (hardest). Within-sector variance is large for top models: MiniMax-M2.7 on Luxury Goods spans 3.6–55.5% Entry-F1 across seeds ($15.2\times$ spread);

Model	En	HI	Lx	Mn	Tr
MiniMax-M2.7	19.6±15.2	35.4±19.2	28.6±23.6	16.9±8.4	44.6±19.9
Qwen3.5-397B	11.1±6.2	10.5±4.4	30.3±15.8	18.5±8.3	32.0±9.9
Qwen3.5-122B	18.0±8.2	24.6±26.9	18.6±10.6	7.2±3.5	30.6±10.0
Qwen3.6-35B	5.0±5.0	10.4±8.9	8.8±10.0	13.2±8.7	11.5±6.1
Mistral-Medium-3.5-128B	8.4±2.9	6.6±4.4	18.5±6.8	5.1±6.3	13.0±7.5
Gemma-4-31B	5.1±2.0	3.2±0.7	13.8±4.3	6.4±1.6	8.7±2.4
Gemma-4-E4B	5.8±5.0	3.6±1.7	4.8±1.9	5.6±8.1	6.4±7.9
GPT-OSS-120B	4.8±3.1	2.8±1.0	7.2±4.0	4.2±2.2	3.9±2.1
Mistral-Small-4-119B	1.8±1.5	0.9±0.8	2.8±2.9	1.7±2.5	4.6±3.7
Qwen3.5-9B	0.8±0.8	2.4±2.9	1.2±2.2	2.3±2.6	3.2±2.3
Granite-30B	0.6±0.8	1.7±1.1	0.9±1.5	0.8±0.6	3.5±1.9
Llama-3.3-70B	0.0±0.0	0.4±0.5	0.0±0.0	0.0±0.0	0.4±0.4

Table 4: Entry Recall (%) by sector.

Model	Fict. acct. payables	Inv.	Rev.	Shd.Pay	Vend.
MiniMax-M2.7	13.1	5.7	40.5	1.6	14.9
Qwen3.5-397B	6.4	5.7	20.2	0.5	13.8
Qwen3.5-122B	2.8	6.5	30.4	1.3	3.3
Qwen3.6-35B	2.3	4.0	14.5	2.8	10.0
Mistral-Medium-3.5-128B	2.0	5.6	19.2	0.0	2.6
Gemma-4-31B	1.3	5.6	13.0	0.0	4.2
Gemma-4-E4B	2.0	0.0	5.6	0.0	1.6
GPT-OSS-120B	0.5	4.1	7.4	0.0	4.0
Mistral-Small-4-119B	0.5	0.0	4.0	1.0	2.2
Qwen3.5-9B	0.8	0.1	2.6	0.0	0.4
Granite-30B	0.2	0.6	2.0	0.0	0.5
Llama-3.3-70B	0.0	0.0	0.0	0.0	0.0

Table 5: Per-scheme recall (%), macro-averaged over 25 runs.

Qwen3.5-122B on Healthcare spans 4.1–74.4% (std 27.0). Multi-seed aggregation is therefore mandatory: single-run scores are unreliable indicators of model quality.

B.3 Per-Scheme Recall

Table 5 reports macro-averaged recall (%) per fraud scheme across all 25 runs. Fictitious account payables disbursements (~ 394 true entries per run) and shadow payroll (~ 56 entries) determine the ranking; revenue manipulation is accessible to all models above 3% Entry-F1.

B.4 Scheme Detection Rate and Investigation Efficiency

Table 6 reports the percentage of 25 runs in which a model achieves non-zero recall for each scheme. Table 7 reports resource consumption (macro-average).

C Failure Analysis

Audit traces from all 300 runs reveal four failure archetypes not visible from aggregate leaderboard scores alone.

C.1 Archetype 1: Breaks the Revenue-Only Trap

Every model above 3% Entry-F1 detects revenue manipulation; the ranking is determined by per-

Model	Fict. acct. payables	Inv.	Rev.	Shd.Pay	Vend.
MiniMax-M2.7	68	24	96	16	60
Qwen3.5-397B	72	16	96	4	52
Qwen3.5-122B	28	20	96	4	24
Qwen3.6-35B	48	8	92	20	48
Mistral-Medium-3.5-128B	40	16	88	0	28
Gemma-4-31B	48	32	100	0	32
Gemma-4-E4B	24	0	64	0	20
GPT-OSS-120B	28	24	84	0	36
Mistral-Small-4-119B	24	0	64	8	12
Qwen3.5-9B	36	4	56	0	8
Granite-30B	16	12	48	0	8
Llama-3.3-70B	0	0	0	0	0

Table 6: Scheme detection rate: % of 25 runs with non-zero recall.

Model	Budg.	Steps	Tkn/Flg	SQLok	Code	Null
MiniMax-M2.7	92	748	66k	99	152	0.0
Qwen3.5-397B	95	738	84k	99	168	0.0
Qwen3.5-122B	61	472	51k	98	64	0.0
Qwen3.6-35B	95	734	169k	99	161	0.0
Mistral-Medium-3.5-128B	73	577	70k	95	80	0.0
Gemma-4-31B	58	533	124k	99	86	0.0
Gemma-4-E4B	42	448	113k	96	86	0.0
GPT-OSS-120B	44	497	162k	99	108	0.0
Mistral-Small-4-119B	62	538	175k	87	195	0.0
Qwen3.5-9B	95	931	355k	99	95	0.0
Granite-30B	26	294	141k	98	19	0.0
Llama-3.3-70B	83	790	787k	100	22	2.6

Table 7: Investigation efficiency. Budg. = % of 20M-token budget used; Tkn/Flg = tokens per flagged entry; SQLok = % SQL calls without error; Code = code-interpreter calls; Null = null-ID flags per run.

formance on fictitious account payables disbursements (~ 394 entries per run, the largest single scheme family; $\approx 41\%$ of fraud volume on a macro-average over sectors). MiniMax-M2.7 is the only model with non-trivial fictitious account payables recall (13.1%; Table 5). It also makes heavy use of the code interpreter (152 calls/run, for Benford tests, amount clustering, and graph analysis beyond SQL screening); however, code-call volume alone does not predict this recall: Mistral-Small-4-119B makes the most code calls (195/run; Table 7) yet reaches only 0.5% fictitious AP recall. We therefore treat the code-interpreter use as a correlate, not a demonstrated cause.

C.2 Archetype 2: Burners with Low Conversion

Qwen3.5-9B (95% budget, 931 steps, 355k tokens/flag), Qwen3.6-35B (95%, 169k/flag), and Qwen3.5-397B (95%, 84k/flag) exhaust the token budget but differ sharply in yield: Qwen3.5-397B achieves $7.5\times$ the Entry-F1 of Qwen3.5-9B at comparable budget usage. **Tokens per flag** (Table 7) is the key diagnostic: high budget plus high Tkn/Flg indicates SQL exploration that does not convert to valid detections.

C.3 Archetype 3: Clean but Shallow

GPT-OSS-120B (44% budget, 63 flags/run, zero null IDs), Gemma-4-31B (58%, 76.1 Consistency), and Granite-30B (26%, 294 steps) terminate early with well-formed output but low recall. GPT-OSS-120B achieves 60.1% revenue precision yet fictitious account payables recall of only 0.5% and shadow payroll recall of 0%; it converges on the salient scheme and declares the investigation complete. Granite-30B stops with 74% of budget unused after minimal investigation; its SQL calls succeed 98% of the time, so the failure is under-investigation rather than query errors.

C.4 Archetype 4: Output-Layer Failure

Llama-3.3-70B (Entry-F1 0.3): workers run 790 steps with 99.9% SQL success, but emit only 26 flags/run at 787k tokens each, including 2.6 null-ID entries per run (document_id: "null", empty rationale). The model misuses report_suspicion to log negative conclusions rather than verified detections, a structured tool-calling failure consistent with its December 2024 release and 50.5% GPQA-Diamond score (Meta, 2024).

Gemma-4-E4B: 45.4% of flags carry scheme_type: unknown, earning zero credit; the 4B model finds anomalies but cannot reliably assign scheme labels.

Mistral-Small-4-119B: 13% SQL error rate (vs. $<2\%$ for top models) compounds over-flagging (precision 13.4%, Consistency 28.5).

C.5 Why Type-F1 Lags Entry-F1

Even MiniMax-M2.7 (Type-F1 21.7 vs. Entry-F1 34.7) correctly labels fewer than one in four detected entries. Scheme classification requires mapping entry-level anomalies to coordinated fraud processes, a capability no model demonstrates at scale. Mistral-Medium-3.5-128B shows the largest gap (Entry-F1 13.6, Type-F1 9.1) due to high-volume misclassification alongside over-flagging.

D Shadow Payroll: The Partial-Access Trap

Shadow payroll is a universal blind spot on ForensicBench: 7 of 12 models score zero recall, and the highest recall is reached by Qwen3.6-35B at only 2.8%; the top leaderboard entry (MiniMax-M2.7) reaches just 1.6% (Table 5). The non-zero shadow-payroll recall of the five remaining models arises from scheme classification of entries already

flagged, not from recovering the blocked HR→GL path: no model independently discovers the structural GL signal (the missing 645100 social-charges line) after `hr_employees` is denied. This section explains the mechanism, documents the compound failure path observed in audit traces, and demonstrates that the scheme is fully detectable via GL structure alone.

D.1 The Trap Mechanism

Every model that reaches the shadow-payroll worker attempts to query the `hr_employees` table within its first two steps; every attempt returns `ERROR: permission denied for table hr_employees`. The table is owned by the database administrator and no `GRANT SELECT` has been issued to the agent user, an intentional design choice modelling a common real-world access restriction (see §3.3).

The failure then compounds in two stages:

Stage 1: Dead-end budget burn. Rather than immediately pivoting to the GL, models spend 60–80% of the shadow payroll worker’s token budget on variations of the inaccessible HR join: attempting `INFORMATION_SCHEMA` workarounds, trying alternative table names, or repeating the same denied query. Audit traces show workers spending 10–17 steps on the `hr_employees` dead end before running out of budget.

Stage 2: No independent GL discovery. The small remaining budget, and more importantly the absence of any GL-structural reasoning in the shadow payroll worker, means models never independently discover the structural payroll signal: a two-line wage accrual posting `Dr 641100 (Salaires) + Cr 421000 (Personnel – rémunérations dues)` *without* a matching social-charges line `645100 (URSSAF/cotisations sociales)`, clustered on the last business day of each month. In legitimate payroll, all three lines appear together: wages, net-pay liability, and employer social charges are posted simultaneously. A ghost-employee payroll entry omits the `645100` line because no real employment contract generates social contributions. This structural test is *entirely derivable* from the French PCG account semantics documented in the schema; no HR data is required.

D.2 Rule-Based Oracle Verification

To confirm detectability, we run an informed oracle that operates under the same database access

Sector	Recall	Precision	n true
Energy	84.2	91.3	38
Healthcare	86.1	88.7	72
Luxury Goods	83.3	93.1	48
Manufacturing	85.0	89.4	60
Transport	83.7	90.2	49
Macro	84.5	90.5	53

Table 8: Informed oracle shadow payroll recall and precision, using GL-structural test only (no `hr_employees` access).

as the agent and applies hand-engineered, scheme-specific SQL tests (Benford-deviation and round-amount filters for fictitious account payables disbursements; period-end reversal detection for revenue manipulation; split-payment graph analysis for vendor collusion; PO-less goods-receipt detection for inventory manipulation). Across the four rule-amenable schemes, the oracle achieves 78–91% recall; for shadow payroll, it applies a GL-structural test: flag any journal entry containing `Dr 641100 + Cr 421000` that lacks a `645100` line, posted on the last business day of the month. The oracle achieves 83–86% shadow payroll recall across the five sectors (Table 8) while operating *without any access to* `hr_employees`: the ghost-employee HR join is never used for the core detections. This confirms that shadow payroll is recoverable from the GL alone; the near-zero LLM recall is not a labelling artifact.

D.3 Design Rationale

The partial-access trap serves three purposes. **(1) Ecological validity:** real forensic investigators routinely encounter restricted HR systems, GDPR-governed tables, or cross-domain access controls that block the “obvious” query path. Adaptive recovery (recognise the denial, identify an equivalent GL signal, pivot) is a core forensic competency. **(2) Capability differentiation:** a model that discovers the GL-structural path after a denied permission query demonstrates qualitatively different reasoning from one that terminates on the error. The current benchmark exposes that *none* of the 12 evaluated models perform this pivot, making shadow payroll the cleanest capability gap in the leaderboard. **(3) Benchmark integrity:** because the scheme is detectable without the restricted table, the near-zero LLM recall cannot be explained by undetectable injections. Future leaderboard submissions that achieve non-trivial shadow payroll

Rank	Model	E-F1	95% Confidence Interval	vs. next
1	MiniMax-M2.7	34.7	[27.7, 41.4]	✓
2	Qwen3.5-397B	26.3	[21.7, 30.6]	overlaps
3	Qwen3.5-122B	25.2	[19.6, 31.7]	✓
4	Qwen3.6-35B	15.0	[10.8, 19.5]	overlaps
5	Mistral-Medium-3.5-128B	13.6	[10.6, 16.7]	overlaps
6	Gemma-4-31B	12.1	[10.8, 13.5]	overlaps
7	Gemma-4-E4B	8.5	[5.7, 12.0]	overlaps
8	GPT-OSS-120B	8.0	[6.4, 9.7]	✓
9	Mistral-Small-4-119B	3.8	[2.4, 5.3]	overlaps
10	Qwen3.5-9B	3.5	[2.1, 5.2]	overlaps
11	Granite-30B	2.8	[1.9, 3.7]	✓
12	Llama-3.3-70B	0.3	[0.1, 0.5]	

Table 9: Entry-F1 (%) with bootstrap 95% confidence intervals over the 25 runs per model (case-level bootstrap, $B = 10,000$; statistic = leaderboard macro-average; point estimates match Table 1 to one decimal). “vs. next” compares the bootstrap difference with the next-ranked model: ✓ = the difference’s 95% confidence interval excludes 0; “overlaps” = the confidence intervals overlap and the pair is not statistically distinguishable. Eight of 11 adjacent entries have overlapping confidence intervals; only 4 of 11 pairwise differences are significant, and the top-2 difference is marginal (95% confidence interval [0.0, 16.6]).

recall provide evidence of genuine adaptive forensic reasoning.

E Uncertainty Quantification and Uninformed Baselines

E.1 Bootstrap Confidence Intervals for the Leaderboard

To quantify leaderboard uncertainty, we compute nonparametric case-level bootstrap 95% confidence intervals over each model’s 25 runs (5 sectors $\times$ 5 seeds), with the leaderboard macro-average (mean over seeds within a sector, unweighted mean over sectors) as the statistic; $B = 10,000$ resamples, fixed seed. For adjacent pairs we bootstrap the difference independently per model.

E.2 Uninformed Standard Audit-Test Baselines

To bound the leaderboard from below with classical (non-agentic) methods, we run a standard *uninformed* audit-test battery over the same public tables the agents query (je_header, je_line): a Benford first-digit test (Benford, 1938; Nigrini, 2012), a round-amount journal-entry test (Lanza, 2006), a period-end clustering test (Lanza, 2006), and an Isolation Forest (Liu et al., 2008) on entry features, plus their union. The battery has no scheme knowledge (no GL-account logic, no access to hr_employees or the label store) and is scored against the same ground truth and protocol

Uninformed audit test	Precision	Recall	Entry-F1
Benford (1st digit $\in \{8, 9\}$)	0.20	9.41	0.38
Round amount (≥ 3 zeros)	0.07	0.18	0.10
Period-end (5 last days)	0.18	30.99	0.36
Isolation Forest (top-1% score)	0.32	1.74	0.54
Audit battery (union)	0.18	37.56	0.36
Best LLM (MiniMax-M2.7, ref. scaffold)	48.4	29.0	34.7
Rule-based oracle (informed)	~ 90	78–91	~ 85

Table 10: Uninformed standard audit-test battery, macro-averaged over the 5 sectors (same ground truth as the leaderboard). Isolation Forest: ROC-AUC 0.757, PR-AUC 0.004 (top-1% operating point shown); the last two rows are reference points from Table 1 and Appendix D.

as the leaderboard.

Two findings follow. First, the battery is a sieve, not a detector: its 37.6% recall is obtained by flagging $\sim 40\%$ of the ledger (608k of 1.51M entries across sectors), yielding 0.18% precision and 0.36% Entry-F1, about $96\times$ lower than the best LLM (34.7%); the top leaderboard score is therefore a result on a genuinely difficult task, not an artefact of the absence of classical baselines. Second, the battery’s recall is carried by the period-end test, which catches period-end-concentrated schemes (revenue manipulation $\sim 55\%$, shadow payroll $\sim 89\%$) while missing the hard cross-table schemes (fictitious AP $\sim 20\%$, inventory $\sim 18\%$, vendor collusion $\sim 24\%$); it assigns no scheme type at all (Type-F1 = 0 by construction), which is precisely the capability ForensicBench evaluates. We note two caveats for interpretation: the period-end filter benefits from the generator’s month-end posting calendar, so part of its recall reflects the synthetic calendar rather than a fraud signal; and the battery runs once per sector (deterministic), against the same static ledgers used for the 5 LLM replicates.

F Fraud Scheme Definitions

Table 12 reproduces the fraud catalogue injected into every agent run (source: fraud_catalogue.py). The catalogue is intentionally conceptual: no GL account codes, posting templates, amount ranges, or calendar cutoffs are disclosed. All concrete indicators must be derived by the agent from its own ledger profiling.

G Agent Prompts

Table 13 summarises the four prompt components assembled at runtime; full source is in the code

Scheme	Process	Multi-entry signature
Fictitious account payables disbursements	Procure-to-pay	Fictitious invoices to shell vendors paid via matched (<i>lettrage</i>) disbursements, or large outflows bounced through a suspense account across shells; account rotation and after-hours posting.
Revenue manipulation	Order-to-cash	Period-end revenue recognised against dummy customers, then reversed in the first days of the next period; concentrated in quarter-end windows.
Vendor collusion	Procure-to-pay	Inflated payments to a colluding or related-party vendor, fragmented into split disbursements sharing one invoice reference; vendor shares bank/address details with an insider.
Shadow payroll	Hire-to-retire	A ghost employee added to master data receives recurring month-end payroll postings (wages, charges, net pay) with no genuine HR lifecycle.
Inventory manipulation	Supply chain	Fictitious goods receipts inflate inventory without purchase orders, followed by year-end write-offs booked as shrinkage.

Table 11: The five injected scheme types. Each spans multiple journal entries across an accounting process; individual postings appear routine, and only their joint structure reveals the scheme.

release. The output contract (below) is part of the system prompt and defines the evaluation unit.

```
{
  "document_id": "<je_header UUID>",
  "entity_id": "<vendor | employee | customer | null>",
  "scheme_type": "<one of five scheme types>",
  "rationale": "One-sentence explanation.",
  "supporting_evidence": ["..."]
}
```

Only `je_header.document_id` UUIDs retrieved from SQL count; invented or null IDs do not score.

Scheme (Process)	Normal process & fraud signals
Fictitious Account Payables Disbursements <i>Procure-to-Pay</i>	Normal: Payables settled only for goods/services ordered, received, and invoiced under segregation of duties; vendor identity and payment references consistent with operating practice. Indicators: Value leaving without a defensible procure-to-pay (P2P) trail; missing receipt or approval evidence; same actors spanning incompatible roles; timing or vendor populations diverging from the rest of the ledger; reference behaviour inconsistent with normal accounts payable (AP) booking.
Revenue Manipulation <i>Order-to-Cash / Record-to-Report (R2R)</i>	Normal: Revenue recognised on delivery; expenses and provisions booked in the appropriate period; reserve releases require defensible economic events. Indicators: Unusual period-end concentration; abnormal account or line mixes for a commercial story; reversals in the first days of the next period; flows inconsistent with order-to-cash (O2C) and R2R activity in this database.
Vendor Collusion <i>(Kickback + Related-Party)</i>	Normal: Procurement follows approved sourcing, pricing, and approval chains; vendors are genuine third parties; no single actor controls selection, invoicing, and payment. Indicators: Inflated or biased spend; repeated coupling of same internal/external parties; payments fragmented to obscure totals; beneficiary or party data overlapping with insiders; weak master-data linkage.
Shadow Payroll <i>Hire-to-Retire</i>	Normal: Payroll settles real employees under HR and payroll policies; accruals and settlements match the entity's payroll design. Indicators: Pay runs structurally odd vs. population in accrual shape, timing clusters, banking patterns, or authorship concentration; recency of HR lifecycle events vs. pay activity. Plausible HR rows do not rule out fraud; master data is context, not a whitelist.
Inventory Manipulation <i>Supply Chain</i>	Normal: Inventory moves trace to economic events (receipts, consumption, authorised adjustments); write-downs require documented physical justification. Indicators: Stock increases without credible upstream support; write-offs disconnected from prior stock history; suspicious pairing of movements; period-close clustering of adjustments; split roles across related postings.

Table 12: Fraud catalogue injected verbatim into the agent system prompt (P2P = procure-to-pay; O2C = order-to-cash; R2R = record-to-report; AP = accounts payable). Agents must derive all concrete GL accounts, SQL filters, and thresholds from ledger profiling, not from this catalogue.

Component	Purpose, Content & Key Design Constraints
System Prompt	Shared static prefix across all phases and workers. Role: <i>senior forensic auditor</i> (COSO, SOX, ISA 240). Injects: (a) database schema, (b) fraud catalogue (App. F), (c) mission, (d) six investigation principles (substance over form; control-circumvention lens; materiality & risk ranking; falsifiability; follow the money; evidence breadth), (e) tool catalogue & preferred workflows, (f) output contract. Catalogue excludes generation parameters (no GL codes, posting templates, or amount parameters).
Orientation Phase	Ledger profiling before any fraud reasoning. <i>No fraud vocabulary</i> permitted (“suspicious”, “anomaly”, “red flag”, “scheme” banned). Eight mandatory structural topics: schema & grain, chart of accounts, process mix, monetary corridors, actor tiers, period structure, master data, controls & references. Output persisted as field notes to <code>orientation_report.md</code> and injected into planning. Default budget: 10% of global token cap.
Planning Phase <i>(one-shot)</i>	Single LLM call emitting <code>investigation_plan.json</code> : a global <code>dispatch_queue</code> (ordered hypothesis tasks across all five schemes) and a per-scheme <code>phases[]</code> array. Each entry carries hypothesis text, a rationale paragraph (claim + benign rival), and a <code>budget_tokens</code> weight. Runtime rescales weights to the post-planning token envelope. Enforces 5–10 distinct hypotheses per scheme; workers dispatched in P-wave order (all P1s before any P2).
Hypothesis Workers	Independent worker agent per hypothesis in a bounded ReAct loop (Yao et al., 2023). Receives task brief (hypothesis, exit criteria, benign rivals, query families). Must emit verdict CONFIRMED / RULED_OUT / INCONCLUSIVE. Calls <code>report_suspicion</code> live as evidence emerges. Writes entity-level findings to a shared <i>blackboard</i> for cross-scheme coordination. All detections merged programmatically; no LLM reconciliation pass.

Table 13: Reference agent prompt architecture. Components assembled at runtime; system prompt is shared by orchestrator and all workers.

Algorithm 1: Multi-stage fraud scheme injection

1. **Require:** Base ledger L (balanced routine entries); calendar days D ; sector config θ ; RNG seed s
2. **Ensure:** Labelled ledger L' ; hidden label store $\mathcal{G}$
3. $S \leftarrow \emptyset$ // active scheme instances
4. $\mathcal{A} \leftarrow \emptyset$ // pending scheme actions
5. $\mathcal{G} \leftarrow \emptyset$ // label records
6. **for** each month m in D **do**
7. **for** each start attempt $k = 1, \dots, K(m)$ **do** // $K(m)$ scales with ledger volume
8. **for** each scheme type $\tau \in \mathcal{T}$ **do** // $|\mathcal{T}| = 5$ in *ForensicBench*
9. **if** Bernoulli($p_\tau(\theta)$) **then**
10. **if** active count of type τ in $S < \kappa_\tau(\theta)$ **then**
11. Select perpetrator u (department-filtered; optional reuse/co-occurrence)
12. Instantiate variant of τ (e.g., salami-slicing vs. expense-laundering for fictitious AP)
13. Register scheme instance s with UUID; $S \leftarrow S \cup \{s\}$
14. **for** each day d in month m **do**
15. **for** each active scheme $s \in S$ **do**
16. $\alpha \leftarrow \text{ADVANCE}(s, d)$ // stage-dependent postings for day d
17. **for** each action $a \in \alpha$ **do**
18. Append a to $\mathcal{A}$; write label to $\mathcal{G}$: (document_id, scheme_type, scheme_instance_id, action_id)
19. **if** s reached terminal stage **then** remove s from S
20. **for** each action $a \in \mathcal{A}$ **do**
21. Materialise a as a balanced journal entry in L (PCG accounts, timestamps, references)
22. Apply forensic markers: fixed created_by, after-hours posting, reference reuse, period-end clustering, suspense routing
23. Update master data from scheme metadata (ghost employees, shell vendors/customers)
24. **return** ($L', \mathcal{G}$)