# Agentic Root Cause Analysis via Hypothesis-Driven Diagnostic Reasoning

**Yusuf Kesmen**
Intelligent Maintenance and Operations Systems (IMOS)
EPFL
yusuf.kesmen@epfl.ch

*Advisor:* Prof. Olga Fink

## 1 Introduction

Modern cyber-physical systems generate continuous sensor streams encoding process health. When anomalies occur, detection systems indicate that *something* has deviated, but rarely reveal *what* has failed or *why*. This gap between detection and diagnosis motivates systematic root cause analysis (RCA) methods.

Consider elevated residuals on sensor $y_3$. The cause could be a bias in $y_3$ itself, drift in a neighboring sensor propagating through shared dynamics, efficiency degradation in an upstream process, or coupling changes between subsystems. Each hypothesis implies different maintenance actions, and misdiagnosis carries significant costs.

Traditional approaches split into model-based methods that leverage analytical redundancy [1, 2] but suffer brittleness to model mismatch, and data-driven methods [3–5] that produce anomaly scores rather than diagnoses. Neither addresses the sequential reasoning required to discriminate among competing hypotheses.

Recent LLM-based agentic systems demonstrate multi-step reasoning with external tools [6, 7]. However, direct application to time-series diagnosis is challenging: raw data is high-dimensional, fault signatures are subtle, and LLMs can produce coherent but incorrect explanations without physical grounding [8]. In high-stakes industrial domains, consistent cumulative evidence gathering is essential—single-shot LLM predictions are insufficient for safety-critical decisions.

This work proposes an agentic RCA framework that structures LLM reasoning through explicit hypothesis tracking. Rather than relying on free-form LLM outputs, we maintain belief scores over a discrete fault space that accumulate evidence across diagnostic steps. Each tool transforms raw time-series into interpretable quantities, and an LLM judge maps findings to discrete evidence scores. We do not model explicit likelihoods; the update is a deterministic log-odds accumulator. This structured approach channels the LLM's reasoning capacity while ensuring consistent, auditable evidence accumulation.

## 2 Related Work

**Model-Based Fault Detection and Isolation.** Classical FDI leverages analytical redundancy through observers, parity relations, and residual generators [9]. The structured residual approach designs generators sensitive to specific fault subsets while insensitive to others [2]. However, these methods require accurate analytical models and face combinatorial complexity as fault modes grow.

**Data-Driven Anomaly Detection.** Autoencoders and VAEs flag high reconstruction error as anomalous [5, 10]; LSTM-based models capture temporal dependencies [11]; graph neural networks encode system structure [12]. However, these methods identify *which* sensor is anomalous but cannot determine *why*—distinguishing sensor faults from propagated process faults requires causal reasoning

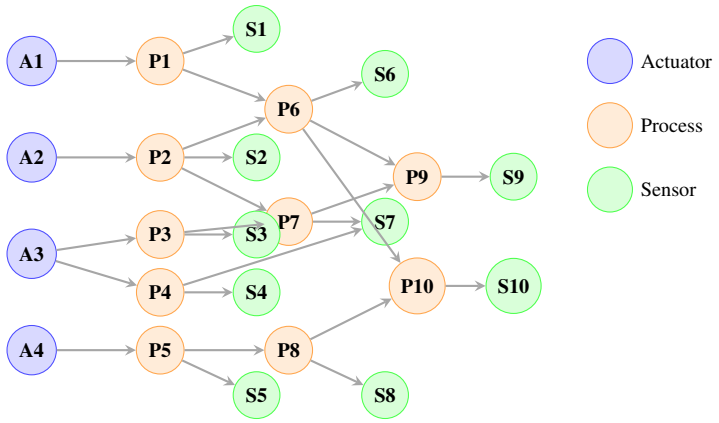


Figure 1: Causal graph $\mathcal{G}$ of the benchmark system. Actuators (blue) drive processes (orange), which propagate through coupled dynamics to sensors (green). Faults at upstream components affect all downstream sensors, enabling causal reasoning for diagnosis.

that data-driven methods cannot learn without labeled fault examples, and GNN-based approaches require retraining for each new system topology.

**Agentic AI and Tool Use.** Chain-of-Thought prompting elicits step-by-step reasoning [13], ReAct interleaves reasoning with actions [6], and Tree-of-Thought explores multiple paths [14]. However, these approaches rely on implicit reasoning over conversation history without structured evidence accumulation. Our experiments show that ReAct-style reasoning achieves high accuracy on sensor faults but fails on process faults where evidence is weak and distributed (Section 5). We address this by combining LLM tool-use with explicit log-odds belief tracking.

**LLM-Based Root Cause Analysis.** RCAgent [15] and GALA [16] apply LLM agents to cloud incident diagnosis via log analysis and service dependency graphs. These focus on discrete event systems; we extend agentic RCA to cyber-physical systems with continuous sensor measurements.

## 3 Problem Formulation

### 3.1 System Model

We consider a discrete-time dynamical system with state $\mathbf{x}_t \in \mathbb{R}^{n_x}$, control inputs $\mathbf{u}_t \in \mathbb{R}^{n_u}$, and measurements $\mathbf{y}_t \in \mathbb{R}^{n_y}$:

$$\mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t) + \boldsymbol{\varepsilon}_t \tag{1}$$

$$\mathbf{y}_t = g(\mathbf{x}_t) + \boldsymbol{\nu}_t \tag{2}$$

where $\boldsymbol{\varepsilon}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{Q})$ and $\boldsymbol{\nu}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{R})$ are process and measurement noise. The causal structure is encoded in a directed acyclic graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, sampled by first drawing a topological ordering of processes and adding directed edges only consistent with this order; hence feedback cycles are excluded.

For the benchmark system, we instantiate $n_u = 4$ actuators, $n_x = 10$ processes, and $n_y = 10$ sensors with coupled autoregressive dynamics:

$$x_{t+1,i} = a_i x_{t,i} + \sum_{j \in \mathrm{Pa}(P_i)} c_{ji} x_{t,j} + b_i u_{t,k(i)} + \varepsilon_{t,i} \tag{3}$$

where $a_i$ is the autoregressive coefficient, $c_{ji}$ the coupling from process $j$ to $i$, and $k(i)$ the actuator driving process $i$. Figure 1 illustrates the causal structure.

### 3.2 Fault Model

A fault is characterized by tuple $\mathcal{F} = (c, \tau, t_0, \theta)$: component $c \in \mathcal{V}$, fault type $\tau \in \mathcal{T}_c$, onset time $t_0$, and severity parameter $\theta$. We consider two fault categories with distinct propagation characteristics.

**Sensor Faults** affect only the measurement channel $y_i$ without altering system dynamics. A *bias* fault applies a constant offset $\tilde{y}_{i,t} = y_{i,t} + \beta$ with $\beta \sim \text{Uniform}[0.5, 1.5]$, producing a step change in the residual mean. A *drift* fault introduces a time-varying offset $\tilde{y}_{i,t} = y_{i,t} + \rho(t - t_0)d$ with rate $\rho \sim \text{Uniform}[0.01, 0.03]$ and direction $d \in \{-1, +1\}$, causing monotonic deviation. A *stuck* fault freezes the measurement at $\tilde{y}_{i,t} = y_{i,t_0^-}$, resulting in variance collapse as the frozen value diverges from dynamic predictions.

**Process Faults** modify system dynamics and propagate through the causal graph. An *efficiency* fault attenuates the autoregressive coefficient $\tilde{a}_i = \gamma a_i$ with $\gamma = 0.5$, reducing response amplitude in process $P_i$ and all downstream sensors $\text{Desc}_{\mathcal{G}}(P_i)$. A *coupling* fault perturbs the inter-process coefficient $\tilde{c}_{ij} = c_{ij} + \delta$ with $\delta \sim \text{Uniform}[0.1, 0.3]$, altering cross-channel correlations without clear localization.

The key diagnostic challenge is that sensor faults produce localized signatures (single channel affected), while process faults produce diffuse signatures (multiple channels with weak individual signals). This asymmetry makes process fault diagnosis inherently harder.

### 3.3 Problem Formulation

The RCA problem requires identifying the active fault from hypothesis space $\mathcal{H}$ comprising $|\mathcal{H}| = 1 + 3n_y + 2n_x$ mutually exclusive hypotheses:

$$\mathcal{H} = \{H_{\text{healthy}}\} \cup \{(S_i, \tau_s) : i \in [n_y], \tau_s \in \mathcal{T}_S\} \cup \{(P_j, \tau_p) : j \in [n_x], \tau_p \in \mathcal{T}_P\} \tag{4}$$

where $\mathcal{T}_S = \{\text{bias}, \text{drift}, \text{stuck}\}$ and $\mathcal{T}_P = \{\text{efficiency}, \text{coupling}\}$.

**Causal Admissibility.** Not all hypotheses are consistent with observed anomaly patterns. Let $\mathcal{S}_{\text{anom}} = \{i : E_i > \theta_E\}$ denote sensors with anomalous residual energy, where $E_i = \sum_t z_{t,i}^2$. The admissible hypothesis space is constrained by the causal graph:

$$\mathcal{C}_{\text{adm}} = \bigcup_{i \in \mathcal{S}_{\text{anom}}} (\{S_i\} \cup \text{Pa}_{\mathcal{G}}(S_i) \cup \text{Pa}_{\mathcal{G}}(\text{Pa}_{\mathcal{G}}(S_i))) \tag{5}$$

This constraint prunes structurally invalid explanations. For example, if only $S_4$ shows anomalous residuals, hypotheses involving $S_1$ faults are inadmissible if $S_1$ does not causally affect $S_4$. This reduces the search space substantially before verification begins.

**Sequential Hypothesis Testing.** Given observations $\{(\mathbf{u}_t, \mathbf{y}_t)\}_{t=1}^T$ and causal graph $\mathcal{G}$, we formulate RCA as sequential hypothesis testing. The agent operates in a loop: at each step $k$, it (1) selects a diagnostic tool based on current beliefs, (2) executes the tool and observes results, (3) updates beliefs based on evidence, and (4) decides whether to terminate or continue. This iterative refinement allows the agent to focus investigation on discriminating between remaining hypotheses.

Let $\pi_k(H)$ denote beliefs at step $k$, initialized uniformly $\pi_0(H) = 1/|\mathcal{H}|$. The design objective is to maximize diagnostic accuracy while minimizing investigation cost:

$$\max \ \mathbb{E}\left[\mathbb{1}[H^* = \hat{H}] - \lambda \sum_{k=1}^{K} c(\tau_k)\right] \tag{6}$$

where $\hat{H} = \arg\max_H \pi_K(H)$ is the final prediction and $c(\tau_k)$ the cost of tool $\tau_k$. Rather than explicitly optimizing this objective, our agent uses LLM-based reasoning to select informative tools and accumulates evidence through a principled belief update mechanism.

## 4 Methodology

The proposed framework comprises four components: a learned world model for residual generation, a diagnostic tool library, a belief tracking mechanism with evidence accumulation, and an agent architecture with in-context learning. Figure 2 provides an overview of the system architecture.

### 4.1 World Model and Residualization

The first stage transforms raw sensor measurements into normalized residuals that highlight deviations from nominal behavior. We train a sequence-to-sequence world model exclusively on healthy episodes,

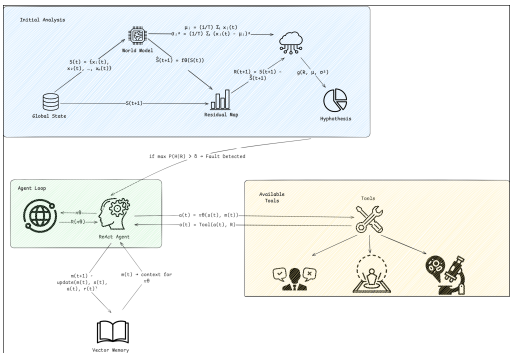

Figure 2: DiagAgent architecture. World model computes residuals, agent loop selects tools and updates beliefs, vector memory stores successful traces for RAP.

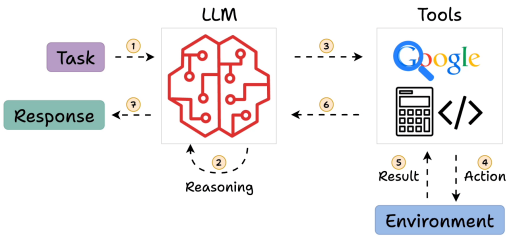


Figure 3: ReAct-style loop. The agent reasons about beliefs (Thought), executes a tool (Action), and updates hypotheses (Observation).

following the principle that a model trained on nominal data will exhibit increased prediction error when presented with faulty data.

The world model takes the form of an LSTM that predicts the next observation given history. Given window length $W$, the input concatenates recent observations:

$$\mathbf{z}_{t-W:t} = [\mathbf{u}_{t-W}, \mathbf{y}_{t-W}, \ldots, \mathbf{u}_t, \mathbf{y}_t] \in \mathbb{R}^{W \times (n_u + n_y)} \tag{7}$$

The LSTM produces hidden state $\mathbf{h}_t$ projected to predict $\hat{\mathbf{y}}_{t+1} = \mathbf{W}_o \mathbf{h}_t + \mathbf{b}_o$. Training minimizes MSE loss on healthy data. At inference, we compute normalized residuals:

$$z_{t,i} = \frac{y_{t,i} - \hat{y}_{t,i}}{\sigma_i} \tag{8}$$

where $\sigma_i$ is estimated from healthy episodes. Under nominal conditions, residuals follow a standard normal distribution; faults cause systematic deviations. Full architecture and training details are provided in Appendix A.2.

### 4.2 Diagnostic Tools: The Verification Layer

The core of our framework is a suite of diagnostic tools that extract structured evidence from residuals. Each tool answers a specific diagnostic question, and the combination enables discrimination between competing hypotheses. The full specification with costs is provided in Appendix A.3.

**Screening tools** provide rapid initial assessment by computing statistics over pre/post-fault windows. The *residual energy* tool ranks sensors by cumulative squared deviation $E_i = \sum_t z_{t,i}^2$, identifying which channels exhibit anomalous behavior. The *variance ratio* tool computes $\text{VR}_i = \text{Var}(y_i^{\text{after}}) / \text{Var}(y_i^{\text{before}})$ to discriminate fault types: $\text{VR} \ll 1$ indicates stuck faults (variance collapse), while $\text{VR} \approx 1$ rules them out. The *mean shift* tool computes z-scores of level changes, where high values suggest bias faults.

**Fault-specific detectors** target individual fault signatures. The *step change test* confirms bias faults by detecting significant mean shifts with preserved variance. The *trend regression* tool fits a linear model to identify drift faults with monotonic temporal patterns. The *variance collapse test* detects stuck faults by checking for near-zero post-fault variance. For process faults, the *response attenuation test* measures reduced autoregressive response, while the *correlation change test* detects altered cross-channel dependencies.

**Causal analysis tools** leverage the system graph $\mathcal{G}$ to distinguish sensor from process faults. The *anomaly spread count* measures how many sensors in $\text{Desc}_{\mathcal{G}}(c)$ exhibit anomalies—high counts support process fault hypotheses since process faults propagate to multiple descendants. The *causal ancestor search* identifies potential root causes by tracing anomaly patterns backward through the causal graph.

Each tool requires calibrated thresholds (e.g., $\theta_E$ for energy, significance levels for statistical tests). We tune thresholds on a held-out validation set to maximize diagnostic accuracy (Appendix A.2).

### 4.3 DiagAgent: Belief Update and Tool Selection

The agent maintains beliefs over all hypotheses $\mathcal{H}$ via log-odds accumulation. After each tool execution, an LLM judge (GPT-4o-mini) assigns evidence scores $s_k(H) \in \{-4, -2, 0, +2, +4\}$ to each hypothesis, representing strong disconfirmation to strong confirmation. Beliefs update as:

$$\ell_{k+1}(H) = \ell_k(H) + s_k(H), \quad P_{k+1}(H) = \frac{\exp(\ell_{k+1}(H))}{\sum_{H'} \exp(\ell_{k+1}(H'))} \tag{9}$$

where $+4$ indicates strong confirmation (e.g., step change with $> 4\sigma$ shift) and $-4$ indicates strong refutation (e.g., normal variance ratio when testing stuck). In practice, a small number of strong confirmations can push the posterior beyond the $0.90$ stopping threshold, especially when competing hypotheses receive weak or negative evidence. Hypotheses with $P(H) < p_{\min}$ are pruned to focus computation.

The diagnostic loop (Algorithm 1 and Figure 6 in Appendix) iterates: (1) the LLM planner selects the next tool based on current beliefs, tool descriptions, and investigation history; (2) the tool executes on residual data; (3) beliefs update via the scoring mechanism above. The loop terminates when $\max_H P(H) \geq 0.90$ or maximum steps $K = 8$ are reached.

**Retrieval-Augmented Planning (RAP).** Rather than training a separate policy, we retrieve in-context demonstrations from successful diagnoses. An experience buffer $\mathcal{B}$ stores successful episodes with their residual signatures $\mathbf{f} \in \mathbb{R}^{n_y}$ (normalized per-channel energy) and tool sequences. For a new episode, we retrieve similar cases via cosine similarity:

$$\mathcal{B}_{\text{similar}} = \arg\max_{b \in \mathcal{B}}^{(k)} \frac{\mathbf{f}^\top \mathbf{f}_b}{\|\mathbf{f}\| \|\mathbf{f}_b\|} \tag{10}$$

Retrieved traces are formatted as natural language demonstrations (e.g., "Similar case $S_3$_bias: residual_energy $\rightarrow$ mean_shift $\rightarrow$ step_change_test"). No parameter learning occurs; we simply provide tool traces as few-shot examples to the LLM planner.

## 5 Experiments

We evaluate our framework on a simulated industrial benchmark with 10 processes and 10 sensors. The evaluation set contains 40 episodes: 8 per fault category (bias, drift, stuck, efficiency, coupling), ensuring balanced coverage across sensor and process faults. Each episode represents a single fault instance; reported accuracies reflect the fraction of correctly diagnosed episodes. Complete experimental configuration is provided in Appendix A.2.

### 5.1 Baselines

We compare against three baseline categories. **Statistical baselines** (Max Residual, PCA-$T^2$, PCA-SPE) identify the most anomalous sensor and assign fault type via variance/mean heuristics; **deep learning baselines** (LSTM-AE, VAE) similarly flag anomalies via reconstruction error. **Agentic baselines**: ReAct [6] uses the same diagnostic tools as DiagAgent but without structured belief tracking—the LLM reasons over raw tool outputs and makes a final diagnosis; Random selects tools uniformly; All-Tools executes all tools exhaustively.

### 5.2 Main Results

Table 1 and Figure 4 present the main comparison. ReAct matches DiagAgent on sensor faults (88%) since both use the same tools, but achieves only 6% on process faults compared to DiagAgent's 69% in our setup. This gap isolates the contribution of structured belief tracking: without accumulating evidence across tools, the LLM cannot integrate weak signals that individually appear inconclusive but collectively indicate a process fault. RAP further improves overall accuracy to 85% by providing effective tool sequences from similar past cases.

### 5.3 Per-Category Analysis

Table 2 breaks down accuracy by fault category. **Drift** and **stuck** faults achieve 100% due to distinctive signatures (monotonic trend, variance collapse). **Bias** is harder (5/8$\rightarrow$6/8) because cascade effects

Table 1: Main results on 40 episodes (8 per category). ReAct matches DiagAgent on sensor faults (88%) but achieves only 6% on process faults vs DiagAgent's 69%—in our setup, tool access without structured belief accumulation is insufficient for weak, distributed evidence.

| **Method** | **Overall** | **Sensor** | **Process** |
|---|---|---|---|
| *Statistical Baselines* | | | |
| Max Residual | 0.25 | 0.42 | – |
| PCA-$T^2$ | 0.20 | 0.33 | – |
| PCA-SPE | 0.28 | 0.46 | – |
| *Deep Learning Baselines* | | | |
| LSTM-AE | 0.30 | 0.50 | – |
| VAE | 0.33 | 0.54 | – |
| *Agentic Baselines* | | | |
| ReAct | 0.55 | 0.88 | 0.06 |
| Random + LLM | 0.12 | 0.17 | 0.06 |
| All-Tools + LLM | 0.20 | 0.33 | 0.00 |
| *Our Method* | | | |
| DiagAgent | 0.80 | 0.88 | 0.69 |
| DiagAgent + RAP | **0.85** | **0.92** | **0.75** |

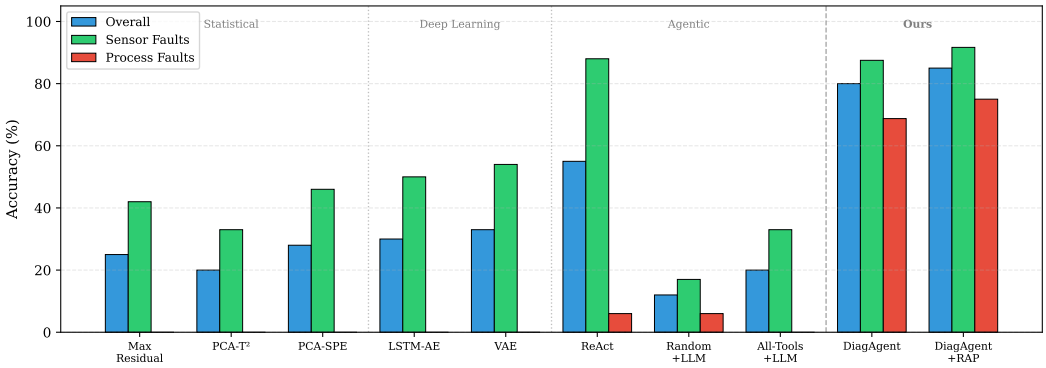


Figure 4: Diagnostic accuracy comparison. Statistical and DL baselines target detection; we include them as reference. ReAct matches DiagAgent on sensor faults (88%) but achieves only 6% on process faults in our setup.

from upstream processes can mimic sensor offsets. **Process faults** achieve 62–75%: efficiency and coupling faults are detected via response attenuation and correlation shifts respectively, but are often confused with each other (Table 3). RAP provides +1 episode improvement on bias and coupling—the most challenging categories.

### 5.4 Diagnostic Trace Analysis

Figure 5 illustrates how belief entropy evolves during diagnosis. Successful cases show steady reduction from 3.93 nats (uniform prior) to below 1.5 nats as each tool provides discriminative evidence. Failed diagnoses plateau above 2.5 nats—the top hypothesis oscillates without converging, indicating ambiguous fault signatures.

## 6 Discussion

**Why Anomaly Detection is Insufficient.** Statistical and deep learning baselines perform anomaly detection, not root cause analysis. They identify *which* sensor deviates from normal but cannot determine *why*—whether the anomaly originates from a sensor fault or propagates from an upstream process. This is not a failure of specific implementations but a fundamental scope difference: these

Table 2: Per-category accuracy (8 episodes each).

| Category | DiagAgent | +RAP |
|---|---|---|
| *Sensor Faults (24 ep)* | | |
| Bias | 5/8 | 6/8 |
| Drift | 8/8 | 8/8 |
| Stuck | 8/8 | 8/8 |
| *Process Faults (16 ep)* | | |
| Efficiency | 6/8 | 6/8 |
| Coupling | 5/8 | 6/8 |
| **Overall** | **32/40** | **34/40** |

Table 3: Confusion matrix (DiagAgent + RAP).

| | Bias | Drift | Stuck | Eff. | Coup. |
|---|---|---|---|---|---|
| Bias | **6** | 0 | 0 | 1 | 1 |
| Drift | 0 | **8** | 0 | 0 | 0 |
| Stuck | 0 | 0 | **8** | 0 | 0 |
| Eff. | 0 | 0 | 0 | **6** | 2 |
| Coup. | 0 | 0 | 0 | 2 | **6** |

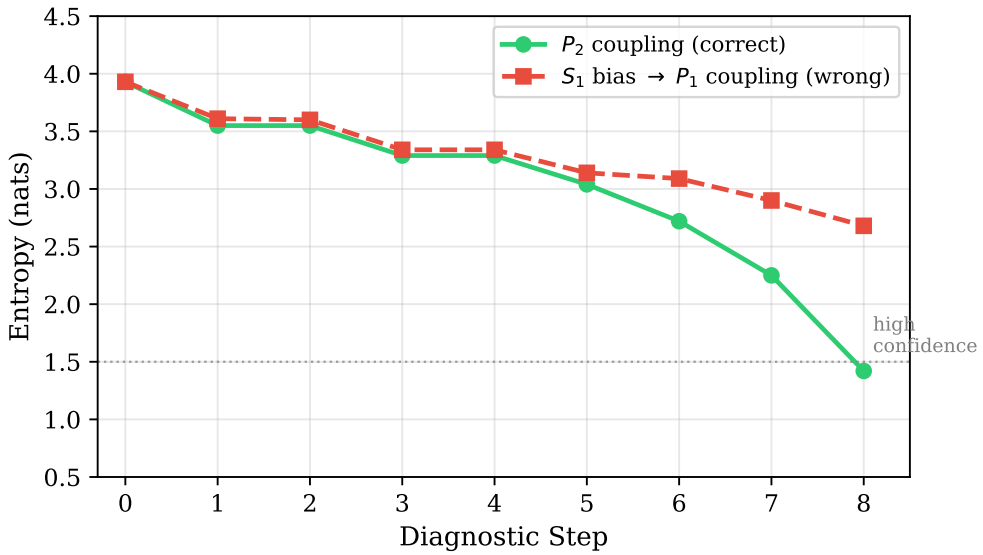


Figure 5: Entropy trajectories for successful (green) and failed (red) diagnoses. Successful cases show steady entropy reduction as tools eliminate hypotheses; failed cases plateau with hypothesis oscillation.

methods are trained on healthy data without fault labels, so distinguishing fault types requires additional reasoning that anomaly scores alone cannot provide.

**Sensor vs Process Fault Asymmetry.** Sensor faults (88%) are substantially easier to diagnose than process faults (69%). This gap stems from signal characteristics: sensor faults produce localized, high-magnitude anomalies in a single channel, while process faults produce diffuse effects across multiple downstream sensors with lower individual magnitudes. The confusion matrix (Table 3) reveals specific failure modes: bias faults are occasionally confused with process faults due to cascade effects, while efficiency and coupling faults show systematic cross-confusion—both produce similar multi-sensor patterns that are difficult to disambiguate even with dedicated detection tools.

**Retrieval-Augmented Planning (RAP).** RAP improves accuracy by 5% (32→34/40) by providing effective tool orderings from similar past cases. The gains concentrate on bias (+1/8) and coupling (+1/8)—the categories where tool sequence matters most. No parameter learning occurs; we simply retrieve similar episodes and provide their tool traces as few-shot examples. This suggests that LLMs can extract procedural knowledge from demonstrations without gradient updates.

**Entropy as Diagnostic Confidence.** Belief entropy provides a reliable indicator of diagnostic certainty. Successful diagnoses show entropy dropping from 3.93 nats (uniform prior) to below 1.5 nats, with sharp drops when detection tools provide confirmatory evidence. Failed diagnoses plateau above 2.5 nats with hypothesis oscillation—the top hypothesis changes multiple times without

converging. This pattern suggests practical stopping criteria: terminate early when entropy drops sharply (high confidence), continue investigation when entropy plateaus (ambiguous evidence).

**Limitations and Future Work.** Several factors limit current evaluation: (1) We use 40 simulated episodes; real industrial processes may exhibit different fault characteristics. (2) The causal graph is assumed known. (3) LLM outputs are stochastic; results may vary across runs. (4) Performance depends critically on the LLM judge mapping tool outputs to discrete evidence scores—we mitigate with low temperature and constrained output schema, but ablation with a deterministic rule-based scorer remains future work. (5) The judge prompts are designed for this fault taxonomy; new fault types require prompt adaptation. (6) Baselines (AE, VAE, PCA) target anomaly detection rather than typed diagnosis; we include them as reference points, not direct competitors. Future work includes validation on real datasets and replacing the LLM judge with learned or rule-based alternatives.

## 7 Conclusion

We presented DiagAgent, a tool-augmented LLM agent for root cause analysis in industrial systems. By combining structured hypothesis tracking with LLM-based tool selection and evidence scoring, DiagAgent achieves 85% diagnostic accuracy with RAP. The key comparison is with ReAct: both use identical tools, yet DiagAgent substantially outperforms ReAct on process faults—demonstrating that structured evidence accumulation is essential when signals are weak and distributed. Our results show that LLMs can perform systematic diagnostic reasoning when augmented with domain-specific tools and log-odds belief tracking.

## References


[1] Rolf Isermann. *Fault-Diagnosis Systems: An Introduction from Fault Detection to Fault Tolerance*. Springer, 2006.

[2] Janos Gertler. Fault detection and diagnosis in engineering systems. 1998.

[3] S. Joe Qin. Survey on data-driven industrial process monitoring and diagnosis. *Annual Reviews in Control*, 36(2):220–234, 2012.

[4] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. Isolation forest. In *Proc. IEEE Int. Conf. Data Mining (ICDM)*, pages 413–422, 2008.

[5] Mayu Sakurada and Takehisa Yairi. Anomaly detection using autoencoders with nonlinear dimensionality reduction. In *Proc. MLSDA Workshop*, pages 4–11, 2014.

[6] Shunyu Yao, Jeffrey Zhao, Dian Yu, et al. ReAct: Synergizing reasoning and acting in language models. In *Proc. Int. Conf. Learning Representations (ICLR)*, 2023.

[7] Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, et al. Toolformer: Language models can teach themselves to use tools. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

[8] Yan Chi, Yichao Fan, Zhen Liu, and Shuicheng Zhou. The mirage of causal reasoning in language models. *arXiv preprint arXiv:2406.04336*, 2024.

[9] Venkat Venkatasubramanian, Raghunathan Rengaswamy, Kewen Yin, and Surya N. Kavuri. A review of process fault detection and diagnosis: Part I–III. *Computers & Chemical Engineering*, 27:293–346, 2003.

[10] Jinwon An and Sungzoon Cho. Variational autoencoder based anomaly detection using reconstruction probability. In *Special Lecture on IE*, volume 2, pages 1–18, 2015.

[11] Pankaj Malhotra, Lovekesh Vig, Gautam Shroff, and Puneet Agarwal. Long short term memory networks for anomaly detection in time series. In *Proc. European Symposium on Artificial Neural Networks (ESANN)*, 2015.

[12] Zhengping Chen, Dongyue Chen, Xiangjie Zhang, Zhanpeng Yuan, and Xueqi Cheng. Learning graph structures with transformer for multivariate time-series anomaly detection in IoT. *IEEE Internet of Things Journal*, 9(12):9179–9189, 2021.

[13] Jason Wei, Xuezhi Wang, Dale Schuurmans, et al. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[14] Shunyu Yao, Dian Yu, Jeffrey Zhao, et al. Tree of thoughts: Deliberate problem solving with large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

[15] Zefan Wang, Zichuan Meng, Jia Liu, et al. RCAgent: Cloud root cause analysis by autonomous agents with tool-augmented large language models. *arXiv preprint arXiv:2310.16340*, 2024.

[16] Yuan Tian et al. GALA: Graph-augmented LLM agents for automated root cause analysis. *arXiv preprint*, 2025.

# A Appendix

## A.1 Algorithm Details

Algorithm 1 presents the complete DiagAgent diagnostic procedure. Figure 6 provides a visual overview of the diagnostic workflow.

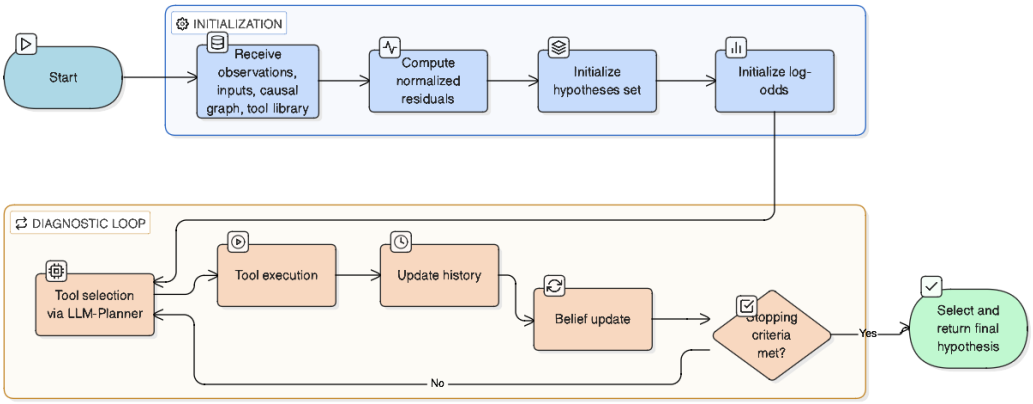


Figure 6: DiagAgent diagnostic workflow. The **Initialization** phase receives episode data, computes normalized residuals via the LSTM world model, initializes the hypothesis set, and sets uniform log-odds. The **Diagnostic Loop** iterates: the LLM-Planner selects the next tool, the tool executes on residual data, history is updated, beliefs are refined via log-odds accumulation, and stopping criteria are checked. The loop terminates when confidence exceeds 90% or maximum steps are reached.

**Algorithm 1** DiagAgent: Hypothesis-Driven Diagnostic Reasoning

**Require:** Observations $\mathbf{Y}$, inputs $\mathbf{U}$, causal graph $\mathcal{G}$, tool library $\mathcal{T}$
**Ensure:** Predicted fault hypothesis $\hat{H}$

$\mathbf{z} \leftarrow \text{LSTM-Residuals}(\mathbf{U}, \mathbf{Y})$
$\mathcal{H} \leftarrow \{H_{\text{healthy}}\} \cup \mathcal{H}_{\text{sensor}} \cup \mathcal{H}_{\text{process}}$
$P_0(H) \leftarrow 1/|\mathcal{H}|$, $\ell_0(H) \leftarrow 0$ for all $H$
history $\leftarrow []$
**for** $k = 1$ to $K$ **do**
  $\tau_k \leftarrow \text{LLM-Planner}(\mathbf{P}_{k-1}, \mathcal{T}, \text{history})$
  $r_k \leftarrow \text{Execute}(\tau_k, \mathbf{z}, \mathcal{G})$
  history.append$((\tau_k, r_k))$
  $\mathbf{s}_k \leftarrow \text{LLM-Judge}(r_k, \mathcal{H})$ {Scores in $\{-4, -2, 0, +2, +4\}$}
  **for** each $H \in \mathcal{H}$ **do**
    $\ell_k(H) \leftarrow \ell_{k-1}(H) + s_k(H)$
    $P_k(H) \leftarrow \exp(\ell_k(H)) / \sum_{H'} \exp(\ell_k(H'))$
  **end for**
  $\mathcal{H} \leftarrow \{H : P_k(H) \geq p_{\min}\}$ {Prune low-probability hypotheses}
  **if** $\max_H P_k(H) \geq p_{\text{stop}}$ **then**
    **break**
  **end if**
**end for**
**return** $\arg\max_H P_K(H)$

## A.2 Experimental Configuration

### A.2.1 Benchmark System

We simulate a coupled dynamical system representing an industrial process with the following components:

Table 4: Benchmark system configuration.

| Component | Description | Value |
|---|---|---|
| Actuators | Control inputs $\mathbf{u}_t$ | 4 |
| Processes | State variables $\mathbf{x}_t$ | 10 |
| Sensors | Measurements $\mathbf{y}_t$ | 10 |
| Episode length | Time steps per episode | 200 |
| Fault onset | Time step when fault is injected | 50 |
| Sampling rate | Simulation time step | 1.0 |

The system dynamics follow coupled autoregressive equations:

$$x_{t+1,i} = a_i x_{t,i} + \sum_{j \in \text{Pa}(i)} c_{ji} x_{t,j} + b_i u_{t,k(i)} + \varepsilon_{t,i} \tag{11}$$

where $a_i \sim \text{Uniform}[0.7, 0.9]$ are autoregressive coefficients, $c_{ji} \sim \text{Uniform}[0.1, 0.3]$ are coupling coefficients, and $b_i \sim \text{Uniform}[0.3, 0.5]$ are input gains. Process noise $\varepsilon_{t,i} \sim \mathcal{N}(0, 0.1^2)$ and sensor noise $\nu_{t,i} \sim \mathcal{N}(0, 0.03^2)$.

### A.2.2 Fault Injection Parameters

Table 5 specifies the fault injection parameters used in data generation.

Table 5: Fault injection parameters by fault type.

| Fault Type | Parameter | Distribution | Description |
|---|---|---|---|
| Sensor Bias | $\beta$ | Uniform$[0.5, 1.5]$ | Additive offset |
| Sensor Drift | $\rho$ | Uniform$[0.01, 0.03]$ | Drift rate per step |
| Sensor Drift | $d$ | $\{-1, +1\}$ | Drift direction |
| Sensor Stuck | $v_{\text{stuck}}$ | $y_{t_0}$ | Frozen at onset value |
| Process Efficiency | $\gamma$ | $0.5$ | AR coefficient multiplier |
| Process Coupling | $\delta$ | Uniform$[0.1, 0.3]$ | Coupling perturbation |

### A.2.3 LSTM World Model Training

The LSTM world model is trained exclusively on healthy episodes to learn nominal system dynamics. Training configuration:

The model takes a sliding window of $W = 50$ time steps of concatenated inputs $[\mathbf{u}_{t-W:t}, \mathbf{y}_{t-W:t}]$ and predicts the next sensor observation $\hat{\mathbf{y}}_{t+1}$. Residuals are computed as $z_{t,i} = (y_{t,i} - \hat{y}_{t,i})/\sigma_i$ where $\sigma_i$ is estimated from healthy validation data.

### A.2.4 Evaluation Protocol

Each evaluation episode proceeds as follows:

1. **Data loading**: Load episode measurements $\{(\mathbf{u}_t, \mathbf{y}_t)\}_{t=1}^{200}$ and ground truth fault label.
2. **Residual computation**: Pass through trained LSTM to obtain residuals $\mathbf{z}_{51:200}$ (post-fault window).
3. **Agent initialization**: Initialize uniform belief over $\mathcal{H}$, reset tool history.
4. **Diagnostic loop**: For up to $K = 8$ steps:
   - DiagAgent selects next tool based on current beliefs
   - Tool executes and returns structured output
   - LLM judge scores hypotheses based on tool output
   - Beliefs updated via log-odds accumulation
   - Check stopping criteria ($p_{\max} \geq 0.90$ or margin $\geq 0.40$)
5. **Prediction**: Return $\hat{H} = \arg\max_H P(H)$

Table 6: LSTM world model training configuration.

| Parameter | Description | Value |
|---|---|---|
| *Architecture* | | |
| Input dimension | $n_u + n_y$ | 14 |
| Hidden dimension | LSTM hidden size | 64 |
| Number of layers | Stacked LSTM layers | 2 |
| Output dimension | $n_y$ (predicted sensors) | 10 |
| Dropout | Between LSTM layers | 0.1 |
| Window size | Input sequence length | 50 |
| *Training* | | |
| Optimizer | | Adam |
| Learning rate | | $1 \times 10^{-3}$ |
| Batch size | | 64 |
| Max epochs | | 50 |
| Early stopping | Patience (val loss) | 10 epochs |
| Loss function | | MSE |
| *Performance* | | |
| Training MSE | Final training loss | 0.0012 |
| Validation MSE | Final validation loss | 0.0015 |
| Training time | On single GPU | $\sim$5 min |

6. **Evaluation**: Record accuracy, steps used, total cost, entropy trajectory

All experiments use GPT-4o-mini with temperature 0.1. Results are averaged over 3 random seeds for LSTM training; the same trained model is used across all diagnostic evaluations.

### A.3 Diagnostic Tool Specifications

Table 7 provides the complete tool library with computational costs. All diagnostic tools accept a query time $t$ as parameter. The agent specifies $t$ to define analysis windows: $\mathcal{W}_{\text{before}}(t) = [t - W, t)$ and $\mathcal{W}_{\text{after}}(t) = [t, t + W]$ where $W$ is the window size (default 40 steps). This formulation allows the agent to probe different time points without assuming known fault onset.

Table 7: Diagnostic tool library with costs. Tools are organized by category: screening for initial assessment, detection for fault-type confirmation, and causal for graph-based reasoning.

| **Category** | **Tool** | **Cost** |
|---|---|---|
| Screening | `residual_energy` | 0.5 |
| | `variance_ratio` | 0.5 |
| | `mean_shift` | 0.5 |
| Sensor Detection | `step_change_test` | 0.8 |
| | `trend_regression` | 0.8 |
| | `variance_collapse_test` | 0.8 |
| Process Detection | `response_attenuation_test` | 1.0 |
| | `correlation_change_test` | 1.0 |
| Causal Analysis | `anomaly_spread_count` | 1.0 |
| | `causal_ancestor_search` | 1.0 |

**Residual Energy Ranking.** Given query time $t$, computes cumulative squared residual per channel in the after-window:

$$E_i(t) = \sum_{\tau \in \mathcal{W}_{\text{after}}(t)} z_{\tau,i}^2 \tag{12}$$

Channels exceeding threshold $E_i(t) > \theta_E$ are flagged as anomalous.

**Variance Ratio Analysis.** Compares variance between windows:

$$\text{VR}_i(t) = \frac{\text{Var}(y_{\tau,i} : \tau \in \mathcal{W}_{\text{after}}(t))}{\text{Var}(y_{\tau,i} : \tau \in \mathcal{W}_{\text{before}}(t))} \tag{13}$$

Interpretation: $\text{VR}_i \ll 1$ suggests stuck fault (variance collapse); $\text{VR}_i \approx 1$ with mean shift suggests bias; $\text{VR}_i \gg 1$ indicates noise inflation or process instability.

**Mean Shift Detection.** Quantifies level change between windows as z-score:

$$Z_i^{\mu}(t) = \frac{\bar{y}_i^{\text{after}}(t) - \bar{y}_i^{\text{before}}(t)}{\hat{\sigma}_i^{\text{before}}(t)} \tag{14}$$

where $\bar{y}_i^{\text{before/after}}$ and $\hat{\sigma}_i^{\text{before}}$ are computed over respective windows.

**Step Change Test.** Confirms bias fault at query time $t$ if mean shift is significant while variance is preserved:

$$|Z_i^{\mu}(t)| > \theta_{\text{bias}} \quad \text{and} \quad \text{VR}_i(t) < \theta_{\text{var}} \tag{15}$$

**Trend Regression.** Fits OLS regression $y_{\tau,i} = \alpha + \rho\tau + \epsilon$ for $\tau \in \mathcal{W}_{\text{after}}(t)$. Drift confirmed if:

$$|\hat{\rho}| > \theta_{\text{slope}} \quad \text{and} \quad R^2 > \theta_{R^2} \tag{16}$$

**Variance Collapse Test.** Confirms stuck fault if variance collapses in after-window: $\text{VR}_i(t) < \theta_{\text{stuck}}$.

**Response Attenuation Test.** Measures downstream response ratio for efficiency faults:

$$\frac{\|\mathbf{y}^{\text{post}}\|_2}{\|\mathbf{y}^{\text{pre}}\|_2} < \theta_{\text{eff}} \tag{17}$$

**Correlation Change Test.** Detects coupling faults via correlation change between linked channels:

$$\Delta\rho_{ij} = \hat{\rho}_{ij}^{\text{post}} - \hat{\rho}_{ij}^{\text{pre}} \tag{18}$$

**Anomaly Spread Count.** Counts anomalous descendants of hypothesized faulty process:

$$N_{\text{anom}}(P_j) = \sum_{S_i \in \text{Desc}(P_j)} \mathbb{1}[E_i > \theta_E] \tag{19}$$

High spread supports process fault; isolated anomaly supports sensor fault.

**Causal Ancestor Search.** For anomalous sensor $S_i$, evaluates consistency for each ancestor $c \in \text{Pa}(S_i) \cup \text{Pa}(\text{Pa}(S_i))$. Returns ranked list of admissible upstream hypotheses.

#### A.3.1 Future Tools (Not Used in Current Experiments)

The following tools are designed but not evaluated in the current experiments. We include their specifications for completeness and future work.

**Counterfactual Analysis.** Estimates residual reduction if hypothesized fault were removed:

$$\Delta E(H) = \frac{E_{\text{affected}}(H) - E_{\text{baseline}} \cdot |\text{affected}(H)|}{E_{\text{total}}} \tag{20}$$

where $E_{\text{affected}}(H)$ is residual energy in sensors affected by hypothesis $H$, and $E_{\text{baseline}}$ is median per-sensor energy. High $\Delta E$ supports the hypothesis as root cause.

**SQL Query Tool.** Provides structured queries over residuals and measurements using SQL-like syntax. Supports operations including: top-$k$ channel ranking by energy, window-based comparisons (pre vs post-fault), aggregations (mean, variance, max), and filtering by threshold. Example: `SELECT channel, energy FROM residuals WHERE t > t0 ORDER BY energy DESC LIMIT 5`.

**Code Interpreter Tool.** Executes custom Python expressions for flexible analysis not covered by predefined tools. Operates in a sandboxed environment with access to numpy functions (mean, std, corrcoef, gradient, etc.) and episode data (residuals $\mathbf{z}$, measurements $\mathbf{y}$, inputs $\mathbf{u}$). Useful for computing custom metrics or testing ad-hoc hypotheses.

### A.4 Threshold Calibration

Detection tool thresholds impact diagnostic accuracy. We tune thresholds on a held-out validation set of 50 episodes to maximize end-to-end accuracy. Table 8 shows the final values, chosen to align with fault generation parameters.

Table 8: Optimized detection thresholds.

| Tool | Parameter | Value |
|---|---|---|
| Residual Energy | $\theta_E$ | 50.0 |
| Mean Shift | $\theta_{\text{bias}}$ | $2.0\sigma$ |
| Variance Ratio | $\theta_{\text{var}}$ | [0.5, 2.5] |
| Drift Slope | $\theta_{\text{slope}}$ | 0.025 |
| Drift $R^2$ | $\theta_{R^2}$ | 0.30 |
| Stuck Detection | $\theta_{\text{stuck}}$ | 0.05 |

The thresholds are calibrated to match fault generation parameters: $\theta_{\text{bias}} = 2.0\sigma$ aligns with bias magnitude $\beta \in [0.5, 1.5]$; $\theta_{\text{stuck}} = 0.05$ captures variance collapse from frozen measurements; drift thresholds match generation rate $\rho \in [0.01, 0.03]$.

### A.5 Hyperparameter Settings

Table 9: Agent and belief update hyperparameters.

| Parameter | Description | Value |
|---|---|---|
| $K$ | Maximum investigation steps | 8 |
| $p_{\text{stop}}$ | Confidence stopping threshold | 0.90 |
| $p_{\min}$ | Pruning threshold | 0.01 |
| $k_{\max}$ | Maximum active hypotheses | 20 |
| $W$ | LSTM window size | 50 |

### A.6 World Model Architecture

The LSTM world model uses two layers with 64 hidden units and dropout 0.1. Input dimension is $n_u + n_y = 14$ (4 actuators + 10 sensors); output dimension is $n_y = 10$. Training uses Adam optimizer with learning rate $10^{-3}$, batch size 64, and early stopping with patience 10 epochs. The model is trained exclusively on healthy episodes to learn nominal dynamics.

### A.7 LLM Configuration

We use GPT-4o-mini for both planning and judging with temperature 0.1 for deterministic outputs. The planner prompt includes: current belief distribution (top-10 hypotheses with probabilities), available tools with descriptions and costs, investigation history (tools called and key findings), and remaining budget. The judge prompt includes: tool name and structured output, active hypothesis list, and scoring guidelines requesting JSON-formatted scores in $\{-4, -2, 0, +2, +4\}$.

### A.8 Computational Cost Analysis

Table 10 compares computational costs across agentic methods. All methods use GPT-4o-mini ($0.15/1M input tokens, $0.60/1M output tokens). Each diagnostic step requires two LLM calls: one for tool selection (planner) and one for evidence interpretation (judge for DiagAgent, implicit reasoning for ReAct).

Table 10: Computational cost comparison (GPT-4o-mini, $0.15/1M input, $0.60/1M output).

| Method | Steps | Input/step | Cost | Acc. | Cost/Corr. |
|---|---|---|---|---|---|
| ReAct | 8.0 | ∼500 | $0.0009 | 52% | $0.0017 |
| DiagAgent | 7.6 | ∼800 | $0.0014 | 80% | $0.0018 |
| DiagAgent + RAP | 7.6 | ∼950 | $0.0016 | 85% | $0.0019 |

**Cost Breakdown.** Each step requires two LLM calls: planner (∼380 tokens) and judge (∼420 tokens). RAP adds ∼150 tokens for few-shot examples. ReAct uses simpler prompts (∼250 tokens/call). DiagAgent's higher per-step cost stems from the detailed scoring rules in the judge prompt, which enable structured belief updates.

**Cost-Effectiveness.** ReAct achieves lowest cost-per-correct-diagnosis ($0.0017) but fails on process faults (6% accuracy). DiagAgent's value lies in diagnosing faults that simpler methods cannot detect—the 69% process fault accuracy justifies the modest cost increase for industrial applications where misdiagnosis is expensive.