FactoryBench: Evaluating Industrial Machine Understanding

Yanis Merzouki[1,2,*] **Coral Izquierdo**[2,3] **Matei Ignuta-Ciuncanu**[4,5] **Marcos Gómez-Bracamonte**[1,6]
Riccardo Maggioni[2] **Alessandro Lombardi**[2] **Camilla Mazzoleni**[2] **Federico Martelli**[2,1] **Balázs Günther**[1]
Jonas Petersen[1,2,†] **Philipp Petersen**[7,†]

[1]ETH Zurich [2]Forgis [3]UC3M [4]Imperial College London [5]University of Berkeley
[6]KTH Royal Institute of Technology [7]University of Vienna
*Correspondence: ymerzouki@ethz.ch †Equal senior contribution

Code: anonymous.4open.science/r/FactoryBench
Dataset: huggingface.co/datasets/FactoryBench/FactoryBench

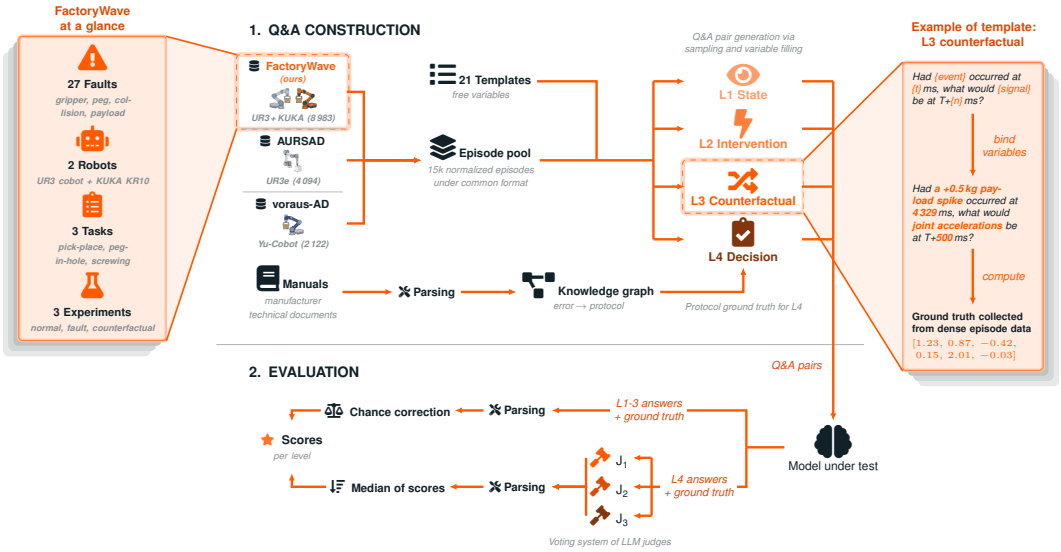


Figure 1: End-to-end FactoryBench pipeline. **(1) Q&A construction:** normalized episodes from FactoryWave, AURSAD [1], and voraus-AD [2] are paired with typed-variable templates (plus the knowledge graph for L4) to generate Q&A items at four reasoning levels. **(2) Evaluation:** the model's answers are scored deterministically for L1–3 and by the median of three LLM judges for L4 free-form, then chance-corrected per level.

Abstract

We introduce **FactoryBench**, a benchmark for evaluating time-series models and LLMs on machine understanding over industrial robotic telemetry. Q&A pairs are organized along four causal levels (state, intervention, counterfactual, decision) instantiating Pearl's ladder of causation, and span five answer formats: four structured formats are scored deterministically and free-form answers are scored by an LLM-as-judge voting protocol. We propose a scalable Q&A generation framework built around structured question templates, present **FactoryWave** (a dense, multitask, multivariate sensor dataset collected from a UR3 cobot and a KUKA KR10 industrial arm), and construct FactoryBench as a large-scale benchmark of over 70k Q&A items grounded in roughly 15k normalized episodes from FactoryWave, AURSAD [1], and voraus-AD [2]. Zero-shot evaluation of six frontier LLMs shows that no model exceeds 50% on structured levels or 18% on decision-making, revealing a wide gap between current models and operational machine understanding.

Preprint.

1 Introduction

Modern robotic manufacturing systems emit dense multivariate telemetry, encoding joint states, torques, forces, velocities, contact events, task phases, and fault indicators. Extracting actionable knowledge from these signals is central to monitoring, diagnostics, anomaly detection, and decision support.

Traditional time-series models excel at narrow tasks such as prediction, classification, and anomaly detection [3–5], but they are typically specialized, hard to interpret, output labels without explicit reasoning, and generalize poorly outside their training task [6–8], limiting their utility for automated engineering decisions in a factory setting. Large language models, in contrast, possess strong general reasoning and can produce structured technical explanations [9, 10], but underperform when applied directly to dense numerical time series [11, 12]; specialized transformer-based architectures meanwhile continue to advance time-series forecasting and representation learning [3, 13–18].

Evaluating whether language-based systems truly reason about machine behavior, rather than pattern-matching on textual cues, remains challenging. By *machine-behavior understanding* we mean the ability to (i) interpret the current operational state from raw multivariate signals, (ii) predict the effect of an intervention, (iii) reason counterfactually about alternative histories, and (iv) recommend remedial actions grounded in physical and engineering constraints. How current LLMs perform across these four competences on dense industrial telemetry has not yet been systematically measured. General benchmarks such as MMLU, BIG-bench, and MMMU [19–21] target textual or multimodal understanding and cannot probe machine behavior; closer-in-spirit time-series Q&A benchmarks (TimeSeriesExam [22], ChatTS [23], EngineMT-QA [24], TSAQA [25], Time-MQA [26], MTBench [27], QuAnTS [28]; Table 1) restrict themselves to synthetic or univariate data, omit counterfactual reasoning, or do not involve a robotic system under closed-loop control.

To address this gap we propose FactoryBench, a benchmark for machine-behavior reasoning in time-series models and LLMs. FactoryBench is grounded in **FactoryWave**, a dense multivariate dataset collected from collaborative and industrial one-arm platforms (UR3 at 125 Hz, KUKA KR10 at 83 Hz) with synchronized setpoint, context, feedback, and effort signals. Industrial-robot data of this kind is rare in public benchmarks: such systems live in restricted production environments behind proprietary controllers with limited telemetry and plant-level confidentiality. To our knowledge FactoryWave is the first extensive anomaly dataset to (partly) cover an industrial robot, explicitly bridging the cobot-to-industrial-machine gap. To populate the benchmark, we introduce a scalable question-generation framework of 21 structured templates spanning state, intervention, counterfactual, and decision-making reasoning, applicable to general machine data flows; instantiated over FactoryWave (and AURSAD / voraus-AD), they yield the full FactoryBench Q&A corpus.

2 Related work

Recent advances in LLM reasoning include prompting and deliberative strategies (e.g., Chain-of-Thought and Tree-of-Thought) that improve multi-step performance [9, 10], alongside tool-augmented paradigms such as Toolformer and ReAct that enable grounded computation through external modules [29–31]. These ideas motivate industrial agent design where symbolic reasoning must be combined with numerical workflows.

Time-series modeling has advanced through transformer-based architectures (Informer, Autoformer, FEDformer, PatchTST) [3, 14–16], strong alternative baselines such as TFT and N-BEATS [32, 33], and critical comparisons against simpler linear models [8]. Foundation-model directions (e.g., Chronos) further explore transfer and zero/few-shot adaptation [17, 18]. In parallel, anomaly and reliability monitoring literature includes OmniAnomaly, USAD, and Deep SVDD [4, 5, 34], with surveys and benchmarks highlighting persistent gaps in interpretability and root-cause actionability [6, 7].

Causal frameworks provide formal tools for interventions and counterfactuals [35–37], while temporal methods such as Granger causality and modern nonlinear causal discovery support directional reasoning in time series [38, 39]. Existing broad benchmarks (MMLU, BIG-bench, MMMU) [19–21] and classical time-series resources [40–42] do not directly evaluate machine-centered reasoning over industrial telemetry. FactoryBench targets this gap by jointly testing temporal interpretation, causal reasoning, and engineering decision support. Table 1 positions FactoryBench against closely related

Q&A and time-series benchmarks along eight axes. The column labels refer to concepts introduced later in the paper: Counterfactual denotes questions that reason about alternative histories (Level 3, Section 3.1); Free-Form denotes open-ended natural-language answers, scored by an LLM-as-judge voting protocol (Appendix A); and Novel Dense TS indicates whether the benchmark contributes a new, high-frequency multivariate dataset rather than relying exclusively on public sources.

Table 1: Comparison of FactoryBench with existing benchmarks. "Counterfactual" = questions reasoning over alternative histories; "Free-Form" = open-ended answers judged by LLMs; "Novel Dense TS" = contributes a new multivariate high-frequency dataset.

Benchmark	Machine/Robot	Multivariate TS	Number of QAs	Counterfactual	Ranking	Numerical	Free-Form	Novel Dense TS
TimeSeriesExam	×	×	∼700	×	✓	×	×	×
ChatTS	×	✓	∼134k	×	✓	✓	×	×
EngineMT-QA	✓	✓	∼110k	×	✓	✓	✓	×
TSAQA	Partial	×	∼210k	×	✓	×	×	×
Time-MQA	×	✓	∼200k	×	✓	✓	✓	×
MTBench	×	✓	∼3k	×	✓	✓	✓	×
QuAnTS	×	✓	∼150k	×	✓	✓	✓	×
CLEVR	×	×	∼1M	×	×	✓	×	×
CLEVRER	×	×	∼305k	✓	×	✓	✓	×
FactoryBench (Ours)	✓	✓	∼71k	✓	✓	✓	✓	✓

3 FactoryBench framework

3.1 Four levels of machine understanding

Table 2: FactoryBench levels, what they test, example questions, and expected answer formats.

Level	Type	Example question	What it tests	Answer format
1	State	"We want to isolate the lifting phase in the robot's time series. Assuming a fixed window length of 15 timesteps, at which timestamp should the window begin?"	Can the model understand what the machine is doing and how it is behaving?	Tensor
2	Intervention	"A collision with a foam cube occurs at $T{=}850$ ms. Rank signal segments (A–D) in the order you would expect them to appear after the event."	Can the model reason about how the machine will behave in the face of an event?	Ranking
3	Counterfactual	"Had a payload misconfiguration occurred at $T{=}200$ ms, what would the target torque on joint 2 at $T{+}50$ ms have been in this counterfactual case?"	Can the model reason accurately about theoretical scenarios?	Scalar
4	Decision	"Given the sensor stream below, does the machine show signs of anomalous behavior? If yes, identify the root cause and the steps to fix it."	Can the model make informed decisions about the machine?	Free-form

Our four-tier hierarchy is explicitly built on top of Pearl's ladder of causation association, intervention, and counterfactual reasoning which has become the organizing principle for causal inference and is increasingly adopted as an evaluation axis for machine-learning systems [35, 36, 43–45]. We extend the causal hierarchy with a fourth decision-making level that reflects how industrial robotic platforms are actually operated: after interpreting state and causal relationships, operators must select and execute corrective procedures, typically prescribed by vendor manuals (e.g., Universal Robots error-code handbooks). This final level aligns with the diagnose-then-act loop that is standard in fault-tolerant control. Grounding the hierarchy in these established principles ensures that each level probes a qualitatively distinct reasoning skill and that failure modes are interpretable in terms of the underlying causal or decision-theoretic primitive.

To systematically evaluate machine-behavior reasoning, we organize question-answering tasks according to this four-tier hierarchy, each probing distinct reasoning capabilities:

Level 1: State. This level assesses the agent's ability to interpret the current state of the machine under normal conditions, including identification of operational modes, prediction and recognition of sensor patterns. Questions at this level require accurate extraction and interpretation of time series features.

Level 2: Intervention. At this level, the agent must reason about the consequences of interventions or events occurring at the present timestep that might perturb the distribution of machine states. Tasks include predicting the immediate impact of control actions, diagnosing faults as they arise, and understanding causal relationships in real time for both normal and anomalous episodes.

Level 3: Counterfactual. This level evaluates the agent's ability to reason about hypothetical scenarios, such as the effect of an event or intervention at a previous timestep. Questions require the agent to simulate alternative histories and assess how outcomes would differ under counterfactual conditions, while still considering the history they know.

Level 4: Decision Making. The highest level evaluates the agent's ability to act on its understanding of the system. Given a recorded episode and a task description, the agent must produce a sequence of actions or recommendations to answer that prompt. In FactoryBench, this targets troubleshooting and optimization, and combines the skills exercised at the previous three levels: reading the system state, reasoning about causes, and weighing alternative interventions. It reflects what we expect from an LLM acting as an engineering assistant on the factory floor.

By structuring Q&A tasks along these four levels, FactoryBench enables rigorous and granular assessment of machine-behavior reasoning, from basic state recognition to advanced decision support. Figure 4 (Appendix B) instantiates Pearl's three causal levels (L1–L3) on robotic time-series data and adds the L4 decision-making layer that FactoryBench introduces on top. Each question is emitted in one of five answer formats (single-select MCQ, multi-select MCQ, ranking, tensor, free-form); the four structured formats are scored deterministically and free-form answers are scored by an LLM-as-judge voting protocol, with per-format details in Appendix A.

3.2 Time series data and causal schema (SCE)

So that the dataset structure itself encodes causality, all time-series signals are organized into three causal groups: **Setpoint** (the controller's commanded target), **Context** (physical and configured conditions, split into static episode metadata and dynamic time-series channels), and **Effort+Feedback** (the machine's measured response). Under healthy operation the response is a known function of setpoint and context; faults manifest as structured residuals from that baseline, giving a single operational fault definition that applies across machines and tasks. Full per-group channel mappings, the formal residual definition, and per-robot calibration details are deferred to Appendix K.

The data conforms to this schema across two source types:

- **FactoryWave:** A custom dataset generated from one-arm robotic platforms executing industrial tasks (e.g., pick-and-place) under varied conditions and systematically injected anomalies.
- **Open Source Datasets:** Adapted datasets such as AURSAD [1] and voraus-AD [2], offering diverse industrial scenarios and preprocessed to conform to the unified episode structure.

The datasets integrated into FactoryBench are summarized in Table 3; all provide the full setpoint–context–effort triplet at 83 Hz or higher and have been (re)labelled to conform to our unified episode schema.

4 Reliable Q&A generation

4.1 Scalable generation via extensive labeling

Scalable ground-truth generation is the central challenge of any Q&A benchmark grounded in raw data. FactoryBench addresses this by coupling a structured labeling ontology with a context-free

Table 3: Datasets incorporated into FactoryBench. FactoryWave is collected and released with this work; the other two are open-source datasets adapted to our schema. Tasks: PnP = pick-and-place, Scr = screwing, PiH = peg-in-hole.

Dataset	Robot	Episodes	Frequency	Tasks	Anomalies
AURSAD [1]	UR3e	4094	100 Hz	Scr	4
voraus-AD [2]	Yu-Cobot	2122	100/500 Hz	PnP	12
FactoryWave (ours)	UR3, KUKA KR10	8983	125/83 Hz	PnP, Scr, PiH	27

grammar (CFG)-style template system, designed by robotics experts and PhD researchers. Each template contains variable slots filled at generation time from one of two sources: label information read directly from the episode data (signal names, timestamps, anomaly labels, root causes), or a predefined sampling distribution attached to that slot (prediction horizons, numerical thresholds, curated vocabularies for discrete options). The discrete vocabularies are seeded from the label sets of AURSAD [1] and voraus-AD [2], extended to cover FactoryWave-specific cases, and released in full so every instantiation is inspectable and reproducible. The context time series used to fill the variables is sampled uniformly by data source (open source vs FactoryWave), then dataset, experiment, length within controlled bounds, and finally placement, yielding a combinatorial expansion from a small set of carefully designed templates into a large, diverse question pool.

Each of the 21 templates is manually authored to probe one specific machine-understanding level while remaining general enough to admit a wide range of instantiations across episode segments, signal names, event descriptions, timestamps, and predicted values. Answer options for single- and multi-select questions are drawn from a shared pool of pre-defined verifiable statements (for example, the question *"What anomaly is present in this robotic sensor time series?"* draws its options from the full catalogue of documented anomalies), each paired with a rule that can be evaluated deterministically against the time series given the densely labelled data, so ground truth is never imputed or inferred but computed directly from the underlying labels; for single-select MCQ items, the option-sampling step additionally enforces that exactly one of the four sampled options evaluates to true on the chosen episode, so the question is unambiguous by construction. This is also our primary defense against the natural concern that template-generated benchmarks can be solved by learning template surface structure rather than the underlying reasoning: every template admits a combinatorially large instantiation space over signal names, timestamps, prediction horizons, payload conditions, and injected-fault types, so two questions sharing the same template rarely overlap in more than a fraction of their variable slots, and the correct answer is never recoverable from the question text alone since it is always determined by the paired time series, which varies across instantiations.

4.2 Density of FactoryWave

FactoryWave is collected from two physical robots (UR3 and the KUKA KR10; Figure 2) executing up to three industrial tasks each: pick-and-place, screwing, and peg-in-hole. Each complete task cycle constitutes one episode, recorded at 125 and 83 Hz across over 100 sensor channels covering joint setpoints, feedback, torques, speeds, estimated contact forces, TCP pose, gripper state, and task-phase labels. To compensate for the limited tool-side telemetry exposed by the KUKA KSS controller, we additionally mount a 9-DoF inertial measurement unit on the KUKA gripper and stream its acceleration, angular-rate, and orientation channels in sync with the controller signals. The full per-robot, per-task, per-condition breakdown of the 8,983 episodes is given in Table 8 of Appendix G.

Episodes are organized into three experiment types. **Normal** episodes capture nominal operation across three payload levels (light, medium, heavy for pick-and-place). **Fault** episodes inject one of 27 fault types (spanning gripper failures, misconfigurations, collisions, and peg-in-hole-specific anomalies), each recorded with fault-specific metadata and, where applicable, a precise injection timestep. **Counterfactual** episodes provide ground truth for Level 3 causal reasoning by approximating the do-operation on physical hardware, via the protocol described next.

Counterfactual generation approximates the last step of Pearl's ladder on physical hardware, which cannot reproduce a bit-identical pre-injection state. For each baseline we re-execute the task 3–5 times with every controllable condition held fixed (object pose, payload, controller configuration,

Figure 2: FactoryWave tasks and fault catalogue. Photographs of the robotic platforms (UR3, KUKA KR10) executing the three tasks (pick-and-place, peg-in-hole, screwdriving) and a representative subset of the 27 systematically injected fault conditions.

exact scripted trajectory) and inject the target fault at a fixed timestep t. We then keep the run whose pre-injection segment minimizes the signature kernel MMD against the baseline, and use that episode as an approximate ground truth for hypothetical divergence of baseline.

4.3 Knowledge graph and protocol extraction

Reliable ground truth for Level 4 troubleshooting questions requires more than episode labels: it demands machine-specific recovery protocols grounded in manufacturer documentation. For each robot in FactoryBench we parse the manufacturer's official runtime error documentation into a structured mapping from error codes to error names, descriptions, and recommended recovery steps, capturing what the controller reports when a fault occurs and what an operator should do to resolve it. We then map every anomaly type studied in FactoryBench to the most likely corresponding manufacturer error code, with PhD-level robotics experts reasoning about the physical cause of each anomaly and identifying which runtime error it would most plausibly trigger on the target robot. Where a physical fault injection (such as gripper misactivation, peg misalignment, or external collision) has no well-defined controller error, the same experts author the recovery protocol directly using the same structure for consistency.

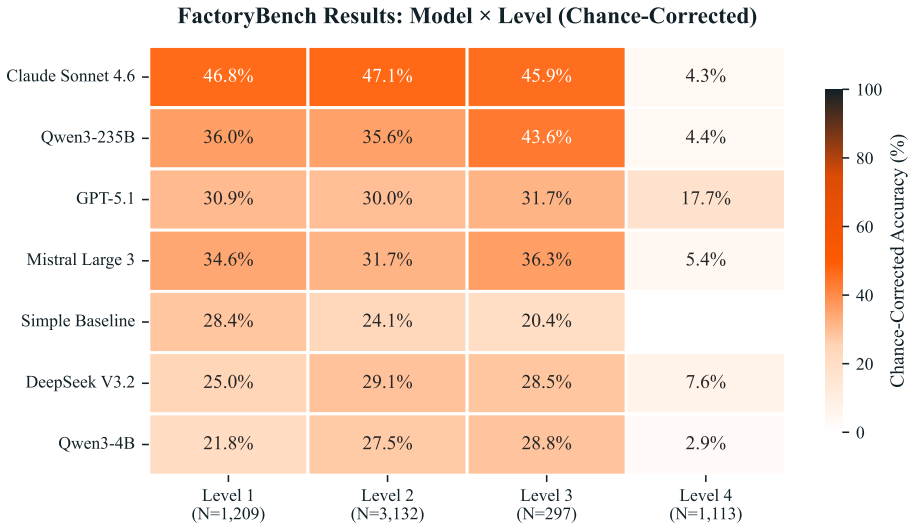
The complete mapping (error-derived and expert-authored protocols alike) is ingested into a knowledge graph alongside robot specifications, task definitions, and signal schema. At question generation time, the pipeline queries this graph to automatically assemble the relevant protocol context and inject it as ground truth into Level 4 troubleshooting Q&A pairs, without requiring per-question human review.

5 Evaluation and results

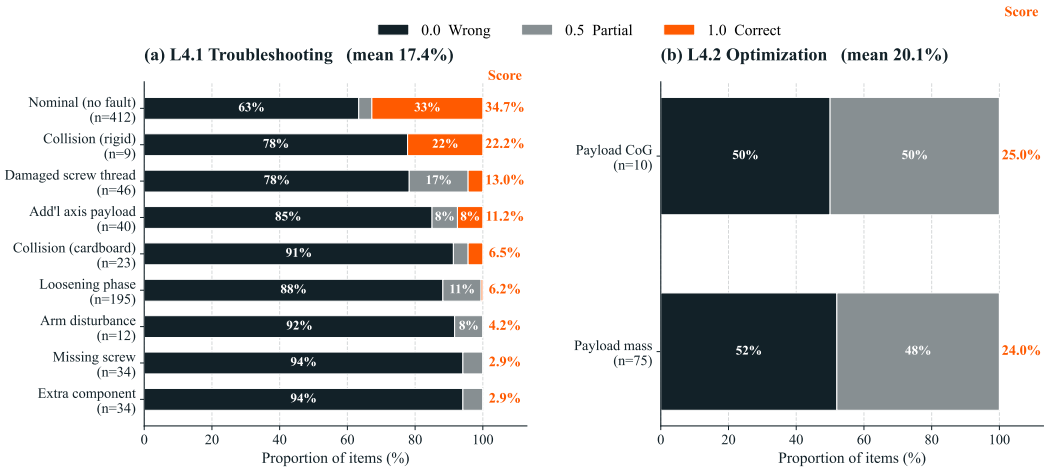
5.1 Zero-shot evaluation

We evaluate a cross-vendor panel of frontier LLMs available at submission time (April 2026): Claude Sonnet 4.6 [46] and GPT-5.1 [47] on the closed-weight side, and DeepSeek V3.2 [48], Mistral Large 3 [49], and Qwen3-235B [50] on the open-weight side. We also include Qwen3-4B [50] in zero-shot mode as a lightweight open-weight reference point. For calibration we report a non-LLM baseline that dispatches per item by answer format: linear regression on the target signal for scalar and tensor templates, and uniform random over the option set for single- and multi-select MCQ and ranking templates. Free-form Level 4 templates are excluded from the baseline (no traditional method maps

cleanly to producing a remediation protocol). Figure 3 reports chance-corrected accuracy for the full panel.



(a) Chance-corrected accuracy (%) on FactoryBench's four reasoning levels.



(b) GPT-5.1 L4 score distribution by ground-truth root cause (left, troubleshooting) and misconfigured parameter (right, optimization). Each bar shows the proportion of items scored 0/0.5/1; the right-edge value is the per-category mean accuracy.

Figure 3: Main results. **(a)** Cross-model panel performance across all four reasoning levels. **(b)** Per-fault and per-parameter breakdown of the GPT-5.1 L4 mass.

5.2 Signal comprehension across Levels 1–3

Most models fail to separate from the simple baseline on Level 1, including the largest in the panel. Three of six LLMs fail to clear the L1 baseline (28.4%): GPT-5.1 (30.9%), by parameter-count and inference-cost proxies the largest in the panel, only matches it, and DeepSeek V3.2 (25.0%) and Qwen3-4B (21.8%) sit clearly below. Only Mistral Large 3 (34.6%), Qwen3-235B (36.0%), and Claude Sonnet 4.6 (46.8%) establish a margin. Raw scale does not reliably confer the ability to extract precise state from dense industrial signals; L1 measures something general-purpose pretraining does not optimize for.

Models that struggle on Level 1 still outperform the baseline on Levels 2 and 3. Qwen3-4B (27.5%, 28.8%), DeepSeek V3.2 (29.1%, 28.5%), and GPT-5.1 (30.0%, 31.7%) all clear the bar on

L2 and L3 despite scoring below it on L1. That being said, the model rank order is nearly identical across L1–L3: Claude Sonnet 4.6 stays in the lead at L2 and L3 (47.1%, 45.9%), Qwen3-235B is the strongest open-weight model and tops L3 (43.6%), and Mistral Large 3 follows (31.7%, 36.3%). This shows a strong correlation between the score of Levels 1–3, and point towards the fact that a stronger state comprehension translates into better intervention and counterfactual reasoning: L2 and L3 build on L1 ability rather than substituting for it.

5.3 Engineering decision-making at Level 4

Level 4 scores collapse absolutely, revealing an industrial readiness gap. Every model collapses below 18%: the leader reaches 17.7% and the other five score between 2.9% and 7.6%. Claude Sonnet 4.6, dominant on L1–L3, crashes to 4.3% on L4, on par with the weakest models in the panel. The model receives the time series and a machine description but must produce the root-cause diagnosis and multi-step recovery procedure from its own knowledge. The L1–3 scores below 50% already indicate a gap in state, intervention, and counterfactual understanding; the L4 collapse exposes a deeper one: models either lack the machine-specific technical knowledge to diagnose faults and prescribe remediation, or cannot ground that knowledge in the observed signals. Closing this gap is the central challenge FactoryBench is designed to measure.

GPT-5.1 leads on the most practically relevant level despite underperforming on the others. The L4 ordering reshuffles completely relative to L1–L3: GPT-5.1, only fourth of six on the structured levels, leads L4 at 17.7%, more than twice the next-best. L4 requires naming the root cause and producing a multi-step remediation procedure grounded in manufacturer documentation; we conjecture that GPT-5.1's reversal reflects disproportionate exposure to structured industrial documentation in pretraining, letting it match a protocol to a fault description even when its signal comprehension is weak. The pattern reveals a dissociation between *signal comprehension* and *protocol retrieval*: optimizing for one does not deliver the other.

Within Level 4, GPT-5.1 shows asymmetric failure modes across troubleshooting and optimization. Breaking GPT-5.1's L4 performance down by question type (Figure 3b) exposes a structural difference in *how* the model fails. On troubleshooting (n=1,011, mean 0.174) the score distribution is sharply bimodal: 79.6% zero, 14.4% perfect, only 5.9% partial. The perfect-score mass concentrates on *nominal* (no-fault) episodes (Figure 3b, left): GPT-5.1 emits "no anomaly" correctly on 33% of healthy episodes, accounting for 92% of all perfects. The remaining 11 perfects fall on collision, payload, and screw-thread faults whose signal signature is strong and unmistakable: a TCP force spike, a speed-scaling collapse, or a large torque drift. Subtler anomalies (task-phase sequencing errors like an unintended loosening step, or flag-level state changes) produce no catastrophic event; GPT-5.1 then over-interprets the signal, attributing the change to a software bug or controller race rather than the expected phase transition, and scores zero or 0.5. Large-n fault categories (loosening phase, missing screw, extra component) sit at $\leq 11\%$ partial and $\leq 1\%$ perfect: even when GPT-5.1 senses that something is off, it cannot recover the canonical root cause from the signal.

On optimization (n=102, mean 0.201) GPT-5.1 *never* achieves a perfect score (41 partial, 61 zero; Figure 3b, right). Ground-truth optimization answers are single-parameter corrections (e.g. "payload mass is set to 0.0 kg but the actual payload weighs 1.5 kg; update the installation settings"). GPT-5.1 instead emits lists of 8–12 generic motion-tuning suggestions (reduce acceleration, retune PID gains, add waypoints, lower speed scaling, enable force-mode compliance) and mentions payload configuration only as one undifferentiated item without specifying direction or magnitude, earning 0.5 at best (according to our grading policy). Together these patterns show that even though GPT-5.1 outperforms every other model by a wide margin on L4, it remains weak at decision-making in an industrial context.

5.4 Modular and extensible by design

FactoryBench is built to grow. The Q&A generator decouples reasoning templates from the underlying robot, task, and signal schema: every template is parameterized over generic primitives (signal names, phase indices, anomaly labels, fault categories) rather than hard-coded to a machine, so adding a new platform only requires populating the labeling ontology and per-robot signal mapping. All existing templates then instantiate automatically, and the format scorers, knowledge graph, and

judge prompts carry over. Planned extensions include further robots and machine families (industrial arms, mobile manipulators, CNC and additive-manufacturing platforms), tasks (assembly variants, polishing, inspection), gripper and end-effector types, and templates targeting under-represented reasoning patterns. Each addition slots into the existing schema without breaking prior versions; the versioned release track pins every reported number to a fixed dataset state while the benchmark continues to grow.

5.5 Limitations

FactoryBench has two substantive limitations. First, Level 3 *counterfactual* ground truth on physical hardware is fundamentally approximate and resource-intensive to generate: two real runs cannot share an identical pre-injection state, so our signature-kernel-MMD-selected CF pair is the closest empirical proxy for the do-operation rather than the do-operation itself, and L3 scores therefore conflate model reasoning with proxy quality (simulation closes the gap only at the cost of physical realism). Second, as the first benchmark of its sort our Q&A pairs use a simplified fault model: industrial faults in production are typically compound, gradual, and detected through aggregate KPIs, while ours are atomic, fast, and drawn from a closed catalogue of 27 injected mechanisms, making FactoryBench a measurement of in-distribution fault *recognition* rather than the fault *detection* problem operators actually face.

6 Conclusion

We introduced FactoryBench, a four-level benchmark for machine understanding over industrial robotic telemetry, grounded in FactoryWave and structured around Pearl's causal hierarchy. Zero-shot evaluation of six frontier LLMs shows that signal comprehension and protocol-grounded decision-making are distinct capabilities that do not co-develop: models that lead on Levels 1–3 collapse on Level 4, while GPT-5.1 reverses rank at L4 despite being mediocre on the structured levels. Absolute scores remain well below 50% on L1–3 and below 18% on L4, with no model close to saturation. A natural follow-up direction is tool-augmented LLM agents that delegate quantitative subtasks to specialised tools (time-series foundation models, classical signal-processing modules, anomaly detectors, or other domain-specific components; Appendix H) and reserve the LLM for the linguistic, classification, and decision-making templates where it has the structural advantage [29–31]. The gap is structural, not incidental: until models can read industrial signals, reason causally about them, and translate that reasoning into operator-grade decisions, they have no business on the factory floor. FactoryBench draws that line, and gives the community the instrument to measure every step taken toward crossing it.

References

[1] Błażej Leporowski, Daniella Tola, Casper Hansen, and Alexandros Iosifidis. AURSAD: Universal robot screwdriving anomaly detection dataset. *arXiv preprint arXiv:2102.01409*, 2021.

[2] Jan Thieß Brockmann, Marco Rudolph, Bodo Rosenhahn, and Bastian Wandt. The voraus-AD dataset for anomaly detection in robot applications. *IEEE Transactions on Robotics*, 40:438–451, 2024. arXiv:2311.04765.

[3] Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 11106–11115, 2021.

[4] Ya Su, Youjian Zhao, Chenhao Niu, Rong Liu, Wei Sun, and Dan Pei. Robust anomaly detection for multivariate time series through stochastic recurrent neural network. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2019.

[5] Julien Audibert, Pietro Michiardi, Frédéric Guyard, Sébastien Marti, and Maria A. Zuluaga. USAD: Unsupervised anomaly detection on multivariate time series. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2020.

[6] Raghavendra Chalapathy and Sanjay Chawla. Deep learning for anomaly detection: A survey. *arXiv preprint arXiv:1901.03407*, 2019.

[7] Alexander Lavin and Subutai Ahmad. Evaluating real-time anomaly detection algorithms—the Numenta anomaly benchmark. In *Proceedings of the IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 38–44, 2015.

[8] Ailing Zeng, Muxi Chen, Lei Zhang, and Qiang Xu. Are transformers effective for time series forecasting? In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2023.

[9] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[10] Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

[11] Nate Gruver, Marc Finzi, Shikai Qiu, and Andrew Gordon Wilson. Large language models are zero-shot time series forecasters. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

[12] Ming Jin, Shiyu Wang, Lintao Ma, Zhixuan Chu, James Y. Zhang, Xiaoming Shi, Pin-Yu Chen, Yuxuan Liang, Yuan-Fang Li, Shirui Pan, and Qingsong Wen. Time-LLM: Time series forecasting by reprogramming large language models. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2024.

[13] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[14] Haixu Wu, Jiehui Xu, Jianmin Wang, and Mingsheng Long. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[15] Tian Zhou, Ziqing Ma, Qingsong Wen, Xue Wang, Liang Sun, and Rong Jin. Fedformer: Frequency enhanced decomposed transformer for long-term series forecasting. In *International conference on machine learning*, pages 27268–27286. PMLR, 2022.

[16] Yuqi Nie, Nam H Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. A time series is worth 64 words: Long-term forecasting with transformers. arxiv 2022. *arXiv preprint arXiv:2211.14730*, 2022.

[17] Abdul Fatir Ansari, Lorenzo Stella, Caner Turkmen, Xiyuan Zhang, Pedro Mercado, Huibin Shen, Oleksandr Shchur, Syama Sundar Rangapuram, Sebastian Pineda Arango, Shubham Kapoor, Jasper Zschiegner, Danielle C. Maddix, Hao Wang, Michael W. Mahoney, Kari Torkkola, Andrew Gordon Wilson, Michael Bohlke-Schneider, and Yuyang Wang. Chronos: Learning the language of time series. *Transactions on Machine Learning Research*, 2024. arXiv:2403.07815.

[18] Abhimanyu Das, Weihao Kong, Rajat Sen, and Yichen Zhou. A decoder-only foundation model for time-series forecasting. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2024. arXiv:2310.10688.

[19] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021.

[20] Aarohi Srivastava et al. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *Transactions on Machine Learning Research*, 2023.

[21] Xiang Yue, Yuansheng Ni, Kai Zhang, Tianyu Zheng, Ruoqi Liu, Ge Zhang, Samuel Stevens, Dongfu Jiang, Weiming Ren, Yuxuan Sun, et al. MMMU: A massive multi-discipline multimodal understanding and reasoning benchmark for expert AGI. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

[22] Yifu Cai, Arjun Choudhry, Mononito Goswami, and Artur Dubrawski. TimeSeriesExam: A time series understanding exam, 2024.

[23] Zhe Xie, Zeyan Li, Xiao He, Longlong Xu, Xidao Wen, Tieying Zhang, Jianjun Chen, Rui Shi, and Dan Pei. ChatTS: Aligning time series with LLMs via synthetic data for enhanced understanding and reasoning. *Proceedings of the VLDB Endowment*, 18(8):2385–2398, 2025.

[24] Yilin Wang, Peixuan Lei, Jie Song, Yuzhe Hao, Tao Chen, Yuxuan Zhang, Lei Jia, Yuanxiang Li, and Zhongyu Wei. ITFormer: Bridging time series and natural language for multi-modal QA with large-scale multitask dataset, 2025.

[25] Baoyu Jing, Sanhorn Chen, Lecheng Zheng, Boyu Liu, Zihao Li, Jiaru Zou, Tianxin Wei, Zhining Liu, Zhichen Zeng, Ruizhong Qiu, Xiao Lin, Yuchen Yan, Dongqi Fu, Jingchao Ni, Jingrui He, and Hanghang Tong. TSAQA: Time series analysis question and answering benchmark, 2026.

[26] Yaxuan Kong, Yiyuan Yang, Yoontae Hwang, Wenjie Du, Stefan Zohren, Zhangyang Wang, Ming Jin, and Qingsong Wen. Time-MQA: Time series multi-task question answering with context enhancement, 2025.

[27] Jialin Chen, Aosong Feng, Ziyu Zhao, Juan Garza, Gaukhar Nurbek, Cheng Qin, Ali Maatouk, Leandros Tassiulas, Yifeng Gao, and Rex Ying. MTBench: A multimodal time series benchmark for temporal reasoning and question answering, 2026.

[28] Felix Wenkel, Ghada Sokar, Yuan Zhang, and Mirco Ravanelli. QuAnTS: Question answering on time series. *arXiv preprint arXiv:2411.04795*, 2024.

[29] Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

[30] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2023.

[31] Yujia Qin, Shengding Hu, Yankai Lin, Weize Chen, Ning Ding, Ganqu Cui, Zhiyuan Zeng, Yujia Huang, Chaojun Xiao, Chi Han, et al. Tool learning with foundation models. *ACM Computing Surveys*, 2024. arXiv:2304.08354.

[32] Bryan Lim, Sercan Ö. Arık, Nicolas Loeff, and Tomas Pfister. Temporal fusion transformers for interpretable multi-horizon time series forecasting. *International Journal of Forecasting*, 37(4):1748–1764, 2021.

[33] Boris N. Oreshkin, Dmitri Carpov, Nicolas Chapados, and Yoshua Bengio. N-BEATS: Neural basis expansion analysis for interpretable time series forecasting. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2020.

[34] Lukas Ruff, Robert A. Vandermeulen, Nico Görnitz, Lucas Deecke, Shoaib Ahmed Siddiqui, Alexander Binder, Emmanuel Müller, and Marius Kloft. Deep one-class classification. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2018.

[35] Judea Pearl. *Causality: Models, Reasoning, and Inference*. Cambridge University Press, 2nd edition, 2009.

[36] Jonas Peters, Dominik Janzing, and Bernhard Schölkopf. *Elements of Causal Inference*. MIT Press, 2017.

[37] Donald B. Rubin. Estimating causal effects of treatments in randomized and nonrandomized studies. *Journal of Educational Psychology*, 66(5):688–701, 1974.

[38] Clive W. J. Granger. Investigating causal relations by econometric models and cross-spectral methods. *Econometrica*, 37(3):424–438, 1969.

[39] Jakob Runge, Peer Nowack, Marlene Kretschmer, Seth Flaxman, and Dino Sejdinovic. Detecting and quantifying causal associations in large nonlinear time series datasets. *Science Advances*, 5(11):eaau4996, 2019.

[40] Nikolay Laptev, Saeed Amizadeh, and Ian Flint. Generic and scalable framework for automated time-series anomaly detection. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2015.

[41] Hoang Anh Dau, Anthony Bagnall, Kaveh Kamgar, Chin-Chia Michael Yeh, Yan Zhu, Shaghayegh Gharghabi, Chotirat Ann Ratanamahatana, and Eamonn Keogh. The UCR time series archive. *IEEE/CAA Journal of Automatica Sinica*, 6(6):1293–1305, 2019.

[42] Qingsong Wen, Tian Zhou, Chaoli Zhang, Weiqi Chen, Ziqing Ma, Junchi Yan, and Liang Sun. Transformers in time series: A survey. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence (IJCAI)*, pages 6778–6786, 2023. arXiv:2202.07125.

[43] Zhijing Jin, Yuen Chen, Felix Leeb, Luigi Gresele, Ojasv Kamal, Zhiheng Lyu, Kevin Blin, Fernando Gonzalez Adauto, Max Kleiman-Weiner, Mrinmaya Sachan, and Bernhard Schölkopf. CLadder: A benchmark to assess causal reasoning capabilities of language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

[44] Nick Pawlowski, Daniel C. Castro, and Ben Glocker. Deep structural causal models for tractable counterfactual inference. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[45] Maxime Peyrard and Robert West. A ladder of causal distances. *arXiv preprint arXiv:2005.02480*, 2020.

[46] Anthropic. Claude Sonnet 4.6 system card, 2026. Anthropic system card, February 2026. `https://www.anthropic.com/claude-sonnet-4-6-system-card`.

[47] OpenAI. GPT-5.1 system card, 2025. OpenAI system card addendum. `https://openai.com/index/gpt-5-system-card-addendum-gpt-5-1/`.

[48] DeepSeek-AI. DeepSeek-V3.2: Pushing the frontier of open large language models, 2025. arXiv:2512.02556. `https://arxiv.org/abs/2512.02556`.

[49] Mistral AI. Mistral Large 3: Model card, 2025. Mistral AI announcement. `https://mistral.ai/news/mistral-3`.

[50] Qwen Team. Qwen3 technical report, 2025. arXiv:2505.09388.

A Answer formats and scoring

FactoryBench emits questions in five answer formats. The four structured formats (single-select MCQ, multi-select MCQ, ranking, tensor) are scored deterministically by per-format rules; free-form answers, used only at Level 4, are scored by an LLM-as-judge voting protocol. Each format is designed to remain hard to answer in the absence of correct reasoning over the time series. Table 4 summarises the raw scoring rules; the rest of this section gives the implementation details together with the chance-correction step that places every format on a comparable scale before aggregation.

Table 4: Answer formats used in FactoryBench and their raw scoring rules. The chance-correction step described in this section is applied on top of these rules before per-level aggregation. Free-form answers (Level 4 only) are scored by an LLM-as-judge voting protocol with three judges (GPT-5.1, Claude Sonnet 4.6, DeepSeek V3.2) aggregated by the median of the three votes.

Format	**Description**	**Raw scoring (before chance correction)**
Single-select MCQ	Three or four mutually exclusive options (A–C or A–D); the model selects one.	Exact match: 1 if the parsed letter equals the gold letter, 0 otherwise. Per-item chance level $E = 1/k$ where $k \in \{3, 4\}$ is the option count for that question.
Multi-select MCQ	Four independent statements (A–D) about the outcome of an event or intervention; the model emits a T/F string of length four.	Positional fraction: $1/4$ per slot whose T/F matches the gold; raw score in $\{0, 0.25, 0.5, 0.75, 1\}$. Chance level $E = 1/2$.
Ranking	Four time-series segments or outcomes (A–D) ordered by a specified criterion. Answer is a permutation string (e.g. `DABC`).	Positional fraction: $1/4$ per slot whose letter matches the gold permutation; raw score in $\{0, 0.25, 0.5, 0.75, 1\}$. Chance level $E = 1/4$.
Tensor	One or more scalar values (e.g. expected sensor reading after an intervention). Answer is a list of floats.	Per-scalar three-level piecewise score: 1 if $\lvert\hat{v} - v^\star\rvert \le m$, 0.5 if $m < \lvert\hat{v} - v^\star\rvert \le 2m$, else 0. Per-channel margin calibrated to $m_j = R_j/12$ so the chance level matches MCQ ($E = 1/4$).
Free-form	Open-ended natural-language response, used only for the two Level 4 question types: troubleshooting and optimization.	Score $\in \{0, 0.5, 1\}$. Troubleshooting: 0.5 if the reply names the correct root cause, 1 if it also gives the correct corrective protocol, 0 otherwise. Optimization: 0.5 if the reply correctly identifies the misconfigured parameter, 1 if it also gives the correct corrective protocol, 0 otherwise.

Chance correction. Without correction, formats with different chance baselines are not directly comparable: a model scoring 0.5 on a multi-select item (chance $E = 1/2$) has demonstrated nothing, while a model scoring 0.5 on a single-select item (chance $E = 1/4$) has demonstrated above-chance performance. We therefore transform every raw item score s into a chance-corrected score

$$\tilde{s} \;=\; \max\!\left(0,\; \frac{s - E}{1 - E}\right),$$

where E is the format's chance level. After this transform, $\tilde{s} = 0$ corresponds to pure-chance performance and $\tilde{s} = 1$ to perfect performance for every format. All level- and template-level means reported in the paper are computed on $\tilde{s}$. The simple non-LLM baseline used in Section 5.1 doubles as the empirical audit: after chance correction it should hover near 0 on every format, and we verify this holds before treating any model score as evidence of reasoning.

Single-select MCQ. Raw score $s \in \{0, 1\}$. The chance level is read off the question payload as $E = 1/k$. The corrected score is $\tilde{s} = \max(0, (k\,s - 1)/(k - 1))$, mapping a correct answer to $\tilde{s} = 1$ and a wrong answer to $\tilde{s} = 0$ regardless of k.

Multi-select MCQ. Raw score $s \in \{0, 0.25, 0.5, 0.75, 1\}$ with chance level $E = 1/2$ (each T/F slot matches the gold with probability $1/2$ under uniform random guessing). The corrected score is

$\tilde{s} = \max(0, 2s - 1)$, mapping $\{0, 0.25, 0.5, 0.75, 1\}$ to $\{0, 0, 0, 0.5, 1\}$. The aggressive flooring is intentional: getting half the slots right under random T/F is the expected outcome of a model that has not read the question.

Ranking. Raw score is the positional-match fraction over a permutation of length four. The expected number of fixed points of a uniformly random permutation is exactly 1 regardless of length, so the chance level is $E = 1/n = 1/4$. The corrected score is $\tilde{s} = \max(0, (4s - 1)/3)$, mapping $\{0, 0.25, 0.5, 0.75, 1\}$ to $\{0, 0, 1/3, 2/3, 1\}$.

Tensor. Each scalar j in a vector answer is scored on a three-level scale: 1 if $|\hat{v}_j - v_j^\star| \leq m_j$, 0.5 if within $2m_j$, 0 otherwise. The raw item score is the average over the n scalars, $s = \frac{1}{n}\sum_j s_j$. The discrete form mirrors MCQ/ranking and still credits near-misses inside $2m_j$. The per-channel margin is calibrated to $m_j = R_j/12$, where R_j is the channel's empirical $[p_2, p_{98}]$ range across the unified episodes (precomputed once); this sets the piecewise scorer's chance expectation to $E = 1/4$, matching single-select MCQ. The corrected item score follows the MCQ form, $\tilde{s} = \max(0, (4s - 1)/3)$. Scalars with a degenerate channel range ($R_j \approx 0$) are dropped from the average.

Free-form. Free-form answers do not admit a meaningful chance baseline: the answer space is unbounded natural language, so there is no uniform random distribution to subtract. We therefore leave free-form scores uncorrected. The lack of correction does not introduce cross-format bias because free-form is used *only* at Level 4, and Level 4 contains no other answer format, so free-form scores are never aggregated alongside structured-format scores at the same level.

The scoring itself is a rubric on $\{0, 0.5, 1\}$, applied against the gold answer assembled from the FactoryBench knowledge graph. The two Level 4 question types are scored separately:

- **Troubleshooting.** The reply receives 0.5 for naming the correct root cause, 1 for naming both the root cause and the appropriate corrective protocol, and 0 otherwise.
- **Optimization.** The reply receives 0.5 for correctly identifying the misconfigured parameter, 1 for identifying both the misconfigured parameter and the general direction even without exact values (increase vs decrease of appropriate dimensions), and 0 otherwise.

Aggregation. Per-question chance-corrected scores $\tilde{s}$ are averaged uniformly within a template and level.

B Question template inventory

Table 5 lists the number of question templates currently producing released Q&A items, broken down by answer format. Templates that are defined in the generator but do not yet emit any questions in the released set are excluded.

Figure 4 instantiates Pearl's three causal levels (Association, Intervention, Counterfactual) on robotic time-series data and adds the Level 4 decision-making layer that FactoryBench introduces on top. Each panel pairs the formal causal primitive with a representative FactoryBench template.

Table 5: Number of active question templates per level, by answer format. Counts reflect templates that produce released Q&A items in the current dataset.

Level	MC single-select	MC multi-select	Ranking	Tensor / Numerical	Free-form	Total
1 (State)	1	1	0	2	0	4
2 (Intervention)	3	2	2	3	0	10
3 (Counterfactual)	0	2	1	2	0	5
4 (Decision)	0	0	0	0	2	2
Total	4	5	3	7	2	**21**

Expert-authored severity ranking. Several Level-2 ranking templates (L2.1, L2.9) require ordering anomalies or signal segments by severity. Severity is not directly readable from the labels:

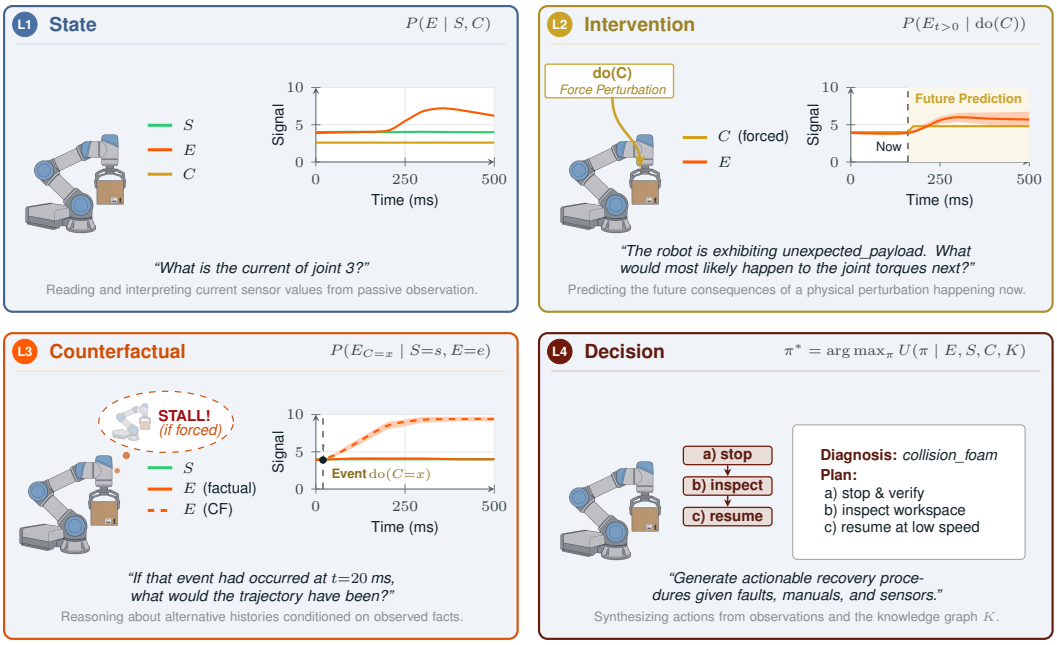


Figure 4: The four levels of machine understanding defined by FactoryBench.

it depends on physical impact, controller reaction, recovery cost, and downstream production consequences. The reference ordering used for these templates was authored and cross-checked by a panel of PhD-level robotics experts, who ranked each anomaly category along these axes and reviewed the resulting ranks for cross-anomaly consistency. The fixed expert ranking is shipped with the benchmark and used as the gold permutation by the deterministic positional-match scorer (Appendix A).

B.1 Full template list

Table 6 lists every active template (the templates that emit released Q&A items in the current dataset). Placeholders in braces ({task}, {anomaly}, {phase}, {signal}, {n}, {event}, {window_length}, {joint_signal}) are filled at generation time from the episode metadata or the relevance specification. The trailing answer-format reminders ("*Answer only with . . .*") are truncated for readability, matching the convention used in Table 17.

Table 6: Full list of FactoryBench question templates by causal level. *Single-choice MCQ* requires one letter; *multi-select MCQ* requires a binary string over a fixed option set; *ranking* requires a permutation over k items; *tensor (scalar)* is a single real number accepted within a tolerance band; *tensor (6-vector)* is a JSON array of six real numbers, one per joint axis; *free-form* responses are evaluated by an LLM-as-judge against a reference answer.

ID	Answer format	Template (boilerplate truncated)
Level 1: State		
L1.1	Tensor (scalar)	The robot is performing a {task} task. We want to isolate the {phase} in the robot's time series. Assuming a fixed window length of {window_length} timesteps, at which timestamp should the window begin?
L1.3	Multi-select MCQ	What changed between the two instances of robotic time series data?
L1.6	Single-choice MCQ	What robot does this sensor data originate from?

continued on next page

Table 6 continued from previous page

ID	Answer format	Template (boilerplate truncated)
L1.7	Tensor (scalar)	Given the sensor stream below, what is the expected value of {signal} at T+{n}ms?
Level 2: Intervention		
L2.1	Ranking	The sensor stream below is from a robot exhibiting {anomaly}. Rank the signal segments listed in the 'options' field in the order you would expect them to appear as the anomaly manifests.
L2.2	Multi-select MCQ	The sensor stream below is from a robot exhibiting {anomaly}. What would most likely happen next?
L2.3	Multi-select MCQ	The sensor stream below is from a robot exhibiting {anomaly}. Select all statements that apply in the options provided.
L2.4	Tensor (scalar)	The sensor stream below is from a robot exhibiting {anomaly}. What is the expected value of {signal} at T+{n}ms?
L2.5	Tensor (6-vector)	The sensor stream below is from a robot exhibiting {anomaly}. What are the expected values of {joint_signal} at T+{n}ms?
L2.6	Tensor (scalar)	Knowing that the robot suffers from {anomaly} in the given context time series, we want to isolate the {phase} phase. Assuming a fixed window length of {window_length} timesteps, at which timestamp should the window begin?
L2.7	Single-choice MCQ	Given the sensor data from a robot performing the task, determine what anomaly is present?
L2.8	Single-choice MCQ	You are provided with two sensor streams originating from robots accomplishing tasks. What differences between the two given instances of robotic time series data (if any) do you notice?
L2.9	Ranking	Rank the following robot time series segments from most to least severe anomaly.
L2.10	Single-choice MCQ	Knowing that the robot suffers from {anomaly} in the given context time series, what robot does this sensor data originate from?
Level 3: Counterfactual		
L3.1	Ranking	Given the baseline sensor stream below and the counterfactual scenario where {event}, rank the signal segments listed in the 'options' field in the order you would expect them to appear as the event manifests.
L3.2	Multi-select MCQ	Given the counterfactual scenario where {event}, what would most likely happen next?
L3.3	Multi-select MCQ	Given the counterfactual scenario where {event}, select all statements that would apply.
L3.4	Tensor (scalar)	In the counterfactual scenario where {event}, what would be the expected value of {signal} at T+{n}ms?
L3.5	Tensor (6-vector)	In the counterfactual scenario where {event}, what would be the expected values of {joint_signal} at T+{n}ms?
Level 4: Decision		
L4.1	Free-form	Given the sensor stream below, does the machine show signs of anomalous behavior? If yes, identify the most likely root cause and describe the steps you would take to fix it.
L4.2	Free-form	An engineer wants to increase the effectiveness and accuracy of this machine. Based on the sensor stream below, what operational changes or parameter adjustments could help achieve this? Do not justify your answer, simply indicate steps to take (or specify if nothing can be done).

Prompt structure and knowledge-graph augmentation. Models under evaluation receive a single user message; no system role is set on the evaluated model. The message is assembled at prompt-build time from four ordered blocks:

```
<machine_sentence>
<acronym_mapping>
<time_series>
Question: <question_text>
Here are the options:
<options>
```

The leading `<machine_sentence>` is filled at prompt-build time from the FactoryBench knowledge graph, specifically the machine and gripper tables, keyed on the dataset that the question is grounded in. This guarantees that identical question templates are grounded in the correct hardware description regardless of which underlying dataset the episode comes from. For a UR3e-grounded question on FactoryWave, the resulting line expands to:

> The following sensor data comes from Universal Robots UR3e, a collaborative robot (cobot) from the E-series with 3 kg payload, 6 degrees of freedom, controlled via UR PolyScope (teach pendant), UrScript, typically used for light assembly tasks, automated workbench scenarios. It is equipped with a 2FG14 Finger Gripper OnRobot.

The augmentation is conditional: identification templates such as L1.6 ("What robot does this sensor data originate from?") and L2.10 suppress the machine sentence so the model cannot read the answer off the prompt header, and templates that probe gripping behavior without revealing tooling can selectively suppress only the gripper clause. The acronym-mapping block decodes the time-series header into long-form channel names and is included whenever the question carries one.

Level-4 free-form responses are scored by an LLM-as-judge under a strict three-point rubric ($\{0.0, 0.5, 1.0\}$) that is template-specific: troubleshooting items (template L4.1) and optimization items (template L4.2) use different prompts, since the relevant axes of correctness differ (root cause + remediation steps vs. problematic parameter + adjustment direction).

Troubleshooting prompt (L4.1).

```
You are scoring a model's troubleshooting answer for a robotics
sensor-data benchmark.

Rubric (apply strictly, no other values allowed):
  1.0  - the answer correctly identifies the underlying root cause AND
         the proposed remediation steps are mostly right. The steps do
         NOT need to match the reference exactly: minor wording
         differences, additional sensible checks, or omitted minor
         steps are all fine, as long as the overall remediation is on
         the right track and would plausibly resolve the issue.
  0.5  - the answer correctly identifies the underlying root cause but
         the remediation is wrong, missing, or off-topic (e.g. proposes
         an unrelated fix, contradicts the reference, or refuses to
         give steps).
  0.0  - the root cause is wrong or absent (model refused, identified
         the wrong fault, or stated "no anomaly" when one was present).

The known root cause is provided to you separately. An answer counts
as identifying the root cause when it names the same physical
phenomenon - equivalent paraphrases are fine, but a different fault
category (e.g. saying "payload too heavy" when the root cause is
"TCP misconfiguration") is wrong.

Respond with ONLY a JSON object on a single line, no markdown:
{"score": <0.0|0.5|1.0>, "reason": "<one short sentence>"}
```

```
Question:        {question}
Reference:       {reference}
Known root cause:{root_cause}
Model answer:    {prediction}
```

Optimization prompt (L4.2).

```
You are scoring a model's optimization answer for a robotics sensor-
data benchmark.

Rubric (apply strictly, no other values allowed):
  1.0  - the answer identifies the problematic parameter AND proposes
         adjusting it in the correct direction (increase vs. decrease,
         matching the reference). Exact magnitudes do not matter; only
         direction matters.
  0.5  - the answer identifies the problematic parameter but proposes
         the wrong direction, no direction, or contradictory directions.
  0.0  - the problematic parameter is not identified, or the model
         recommends adjusting an unrelated parameter, or refuses to
         answer.

The reference answer states the correct parameter and its target
value, from which the correct adjustment direction can be inferred
relative to the configured value.

Respond with ONLY a JSON object on a single line, no markdown:
{"score": <0.0|0.5|1.0>, "reason": "<one short sentence>"}

Question:    {question}
Reference:   {reference}
Model answer:{prediction}
```

The judge ensemble comprises three frontier LLMs from different vendors: **GPT-5.1**, **Claude Sonnet 4.6**, and **DeepSeek V3.2**. Each judge independently scores every Level-4 free-form response with the appropriate template-specific prompt above, and the per-item score reported in our results is the median of the three valid votes (snapped to the $\{0,\ 0.5,\ 1\}$ grid).

C Episode eligibility filtering

Before any sub-window is drawn (Appendix D), each template restricts the pool of source episodes to those that can support a well-defined ground-truth answer. The filter is template-aware and combines a level-specific condition predicate with a length/coverage check.

Level-specific condition predicate. The level a template lives at fixes which episode condition the template can pull from: Level 1 templates probe nominal state and therefore consume only *baseline* (normal) episodes, Level 2 templates probe the immediate effect of an intervention and therefore consume only *anomalous* (fault) episodes, and Level 3 templates probe counterfactual reasoning and therefore consume only *counterfactual* pairs (one baseline plus one fault run, paired by the protocol described in Appendix G). Level 4 templates draw from baseline and fault episodes depending on the troubleshooting/optimization variant. Episodes whose condition does not match the template are removed from the pool before sampling.

Ground-truth feasibility. Eligible episodes must additionally carry the labels and trajectory length the template needs to derive its answer. Concretely, we require that (i) the episode contains every metadata field the template instantiates (root-cause label, anomaly type, fault-injection timestep, target phase, etc.), (ii) the episode is long enough that a relevance-compliant sub-window of the requested length (Appendix E) fits, and (iii) for templates whose answer is computed from a future segment of the trajectory (numerical and tensor forecasts at L1.7, L2.4, L2.5), the trajectory extends at least the requested forecast horizon beyond the displayed window. Episodes that fail any of these

checks are dropped at filter time rather than at sampling time, so each template's sample distribution is over its true eligibility pool.

Effect on the released set. The filter is applied per (template, dataset, condition) cell before the train/validation/test split, so the public 70k Q&A items released on Hugging Face are exactly the items for which a verifiable ground-truth answer exists.

D Fault-relevance window sampling

For Q&A pairs grounded in a fault episode, the time-series context shown to the model is a sub-window of the full episode rather than the entire trajectory. A naive uniformly sampled sub-window can miss the segment where the fault is actually visible: for a transient collision lasting tens of milliseconds, a uniformly sampled one-second window has a non-negligible probability of falling entirely outside the disturbed segment, producing an item whose ground-truth label is correct but whose evidence is invisible to any model. To prevent this, each fault type is paired with a *relevance specification* that constrains how the sub-window is drawn so that the fault signature is, by construction, present in the displayed context.

Four temporal footprints. Each fault is assigned, by domain annotation, to one of four classes that describe how its signature is distributed across the episode:

- *Global*: the signature is present throughout the entire episode, for instance a payload misconfiguration that biases every torque reading. Sub-windows are sampled uniformly under the requested length bounds.
- *Event-anchored*: the signature is a transient localized at a specific timestep recorded in the episode (collisions, gripper misactivation). The sub-window is required to contain that timestep; its length and start are otherwise random.
- *Phase-gated*: the signature is only diagnostic during specific task phases, for instance a peg-misalignment fault that is only visible during insertion. The sub-window is required to overlap one of the target phases by at least a minimum number of rows, where the relevant phases depend on the task.
- *Cumulative*: the signature has a monotonically growing footprint that becomes visible only after a certain phase begins, for instance progressive torque drift. The sub-window is required to be at least a minimum length and, where applicable, to start no earlier than the onset phase.

Sampling and validation. Given a fault episode and the requested length bounds, the sampler draws a window that satisfies the corresponding constraint and records the constraint that was applied alongside the question's provenance, so downstream analyses can condition on it. A post-hoc check verifies that the returned window meets the locality's requirement; if it cannot be met (e.g., a phase-gated fault whose target phase is missing from the recording), the item is either discarded or, for templates where the anomaly signature is helpful but not required for correctness, the sub-window falls back to uniform sampling and the item is flagged accordingly.

Effect on the dataset. Relevance-aware sampling is applied to all fault-grounded items in Levels 1, 2, and 4. Level 3 counterfactual pairs use a separate fixed-injection-timestep protocol described in Section 4.2 and do not rely on this mechanism. In pilot generation runs without relevance constraints, a non-trivial fraction of fault items had sub-windows that did not contain the fault signature, inflating the apparent difficulty of Levels 1 and 2 in a way that could not be distinguished from genuine model failure. The mechanism can be disabled to recover uniform sampling for ablation studies that require a no-relevance baseline.

E Time-series context: features and window length

Two design choices govern what each Q&A item actually shows the model: *which* channels of the multivariate trajectory are included, and *how long* the displayed window is. Both choices are deliberately constrained.

Resampling to a uniform 10 Hz. The source datasets record at heterogeneous frequencies (83–125 Hz for FactoryWave, 100 Hz for AURSAD [1], 100 or 500 Hz for voraus-AD [2]). Before any Q&A item is built, every episode is anti-alias filtered and resampled to a common 10 Hz timeline. This serves three purposes: it normalizes the time axis across robots so that templates can specify window lengths in absolute seconds without conditioning on the source platform, it keeps the displayed context within the prompt budget of every evaluated model, and it removes high-frequency content that would otherwise dominate the token sequence with information not needed for the reasoning levels we evaluate. Ten Hertz is admittedly low by industrial-sensor standards: at higher sampling rates one can resolve fast transients (millisecond-scale collisions, valve switches) that are smoothed away here. We accept this trade-off because the fault catalogue evaluated in this paper is deliberately simplified to atomic, well-isolated mechanisms, and we observe empirically that 10 Hz is sufficient to expose every fault signature in our catalogue at a level that PhD-level annotators can verify on the displayed context. Future versions of the benchmark targeting compound or sub-second faults would need to revisit this choice.

Per-template feature pre-selection. The raw signal schema is large: FactoryWave exposes more than 120 channels per timestep (joint positions, velocities, accelerations, currents, target torques, contact forces, TCP pose, gripper state, and so on). Including every channel verbatim would produce contexts that are both prohibitively long and dominated by signals irrelevant to the question being asked. We therefore equip every question template with a hand-curated subset of *important features*: 30 channels per template for Levels 1–3, and roughly 20 for Level 4 (whose templates focus on the diagnostic and remediation signals that matter for troubleshooting). The selections are authored by PhD-level robotics experts on the team and target the channels most informative for the specific reasoning skill the template probes; for instance, an anomaly-detection template emphasizes torques and contact forces, while a phase-isolation template foregrounds joint positions and velocities. The full per-template feature lists are released alongside the benchmark so that every Q&A item can be reproduced bit-for-bit from the underlying episode.

Window length. Across all four levels, each Q&A item shows a sub-window of between 32 and 64 timesteps drawn from the 10 Hz-resampled trajectory, corresponding to roughly 3.2–6.4 s of robot motion. The lower bound is set so that even the shortest displayed window contains enough samples to expose the dynamics the templates probe (transient onsets, phase boundaries, force spikes); the upper bound keeps the prompt within the context limits of all evaluated models, including the smaller open-weight ones. Within these bounds, the exact length and start position are randomized at generation time, with the start position constrained by the relevance specification of Appendix D when the item is fault-grounded.

Why these choices matter for interpretability. Fixing the resampling rate, the feature subset, and the window-length range across templates means that performance differences between models on a given template cannot be attributed to one model receiving a longer, faster, or richer context than another, only to its ability to interpret the same content. The randomization within bounds prevents models from exploiting positional or length cues to recognize templates by their surface form: two items instantiated from the same template are typically presented with different lengths, different start positions, and – once relevance constraints and per-template feature subsets are honoured – with the fault evidence falling at different relative positions in the displayed window.

F Validation of solvability

This appendix collects the diagnostic checks we ran to convince ourselves that FactoryBench items are answerable from the released time-series context, rather than from question wording alone, and that the templates flagged as "too easy" by the noise-substitution check are nevertheless grounded in genuine signal structure.

F.1 Noise-substitution check

To probe whether the model scores in Figure 3 reflect actual reading of the time-series context or merely priors over question templates and option positions, we constructed a noise-substituted counterpart of the dataset and re-ran every model against it. This check was an early diagnostic study

run on a previous, smaller model panel (Claude Haiku 4.5, DeepSeek V3.1, gpt-5.1, Mistral Large 3) and a 400-item Q&A subset; it predates the main evaluation reported in the body of the paper, so the absolute scores below are not directly comparable to those in Figure 3. In particular, this early subset was generated with substantially longer time-series contexts than the released benchmark, which depresses scores across the board for every model in the panel; the relative differences between Orig and Sub remain interpretable, but absolute levels run roughly $5-10$ chance-corrected points below the comparable cells in the main results.

Construction. For each of the 400 generated Q&A pairs (100 per level), we duplicate the item and rewrite only the numerical time-series content. For each numeric *float* feature we preserve the original first value and substitute every subsequent value with the cumulative sum of independent Gaussian increments:

$$v'_0 = v_0, \qquad v'_{t+1} = v'_t + \mathcal{N}(0, \sigma^2), \qquad \sigma = 0.5,$$

where σ is shared across all features. The same substitution is applied to time-series chunks embedded in option strings (Level 1 severity-ranking, Level 2/3 chunk-ranking).

Expected behavior. A model that scores by genuinely reading the time series should drop sharply on the noise-substituted variant, because the substitution destroys the signal-trajectory information that the gold answer hinges on. A model that scores by template-level priors (e.g. guessing the most plausible MC option given the question wording) or by option-position bias should be roughly invariant, since the question text and options are unchanged.

Result. Table 7 reports the side-by-side mean rubric scores. Three patterns emerge.

1. On Level 1 every model is essentially flat or improves slightly under substitution. This is consistent with L1 templates being substantially answerable from the question wording, the option set, and the still-present discrete annotations of task phase and fault label. It also flags L1 as the level where overlap between LLM scores and a random/regression baseline is most expected.
2. On Level 2 and Level 3 every model drops, often substantially: DeepSeek and Mistral lose 9–11 points on L3, and gpt-5.1 loses 7.7 on L2. The L3 drops are particularly informative because L3 templates ask counterfactual "what would have happened" questions where the answer hinges on the actual signal trajectory rather than on a discrete annotation.
3. On Level 4, gpt-5.1's 7.6% original score collapses to 0.5% under noise substitution, a $15\times$ drop. Inspection of the per-item rubric notes shows that gpt-5.1's L4 mass on the original data came almost entirely from correctly emitting "no anomaly is present" on truly nominal episodes; once the time series is replaced by Gaussian noise, the same nominal episodes look anomalous to the model and it now hallucinates faults on them, losing its only mode of scoring. The other three models score ≤ 0.005 in either condition and have no comparable mode to lose.

Table 7: Noise-substitution check: measured mean scores (%) on the original FactoryBench Q&A pairs (Orig) vs. the noise-substituted variant (Sub) constructed by replacing every numerical time-series value with cumulative Gaussian increments seeded at the original first value ($\sigma = 0.5$). Questions, options, and gold answers are identical between the two runs. Δ = Sub − Orig; large negative values are evidence that the model was using the data from the time series rather than the question alone.

	L1			L2			L3			L4		
Method	Orig	Sub	Δ	Orig	Sub	Δ	Orig	Sub	Δ	Orig	Sub	Δ
Claude Haiku 4.5	19.0	19.5	+0.5	25.5	26.7	+1.2	6.3	4.4	-1.9	0.5	0.0	-0.5
DeepSeek V3.1	16.5	15.0	-1.5	23.7	19.5	-4.2	24.7	15.7	-9.0	3.5	4.0	+0.5
gpt-5.1	24.0	24.0	0.0	28.7	21.0	-7.7	22.0	16.0	-6.0	7.6	0.5	-7.1
Mistral-Large-3	22.0	25.0	+3.0	22.7	21.8	-0.9	26.7	15.8	-10.9	0.0	0.0	0.0

F.2 Distribution-shift analysis

For the templates that produced a small noise-substitution Δ in the previous subsection, we confirm that they are nevertheless solvable by further empirical analysis, for example by inspecting how the sensor data distribution shifts when a fault is present. Figure 6 illustrates this by comparing multiple healthy KUKA episodes (blue) against multiple faulty episodes with an arm-disturbance fault (orange) drawn from FactoryWave (Figure 5 shows the physical setup of the fault for visual intuition), and reveals a clear distributional shift between the two groups on the channels the affected templates read from. The shift is more pronounced on some joints than on others, as the angle at which the disturbance is applied loads them unequally.

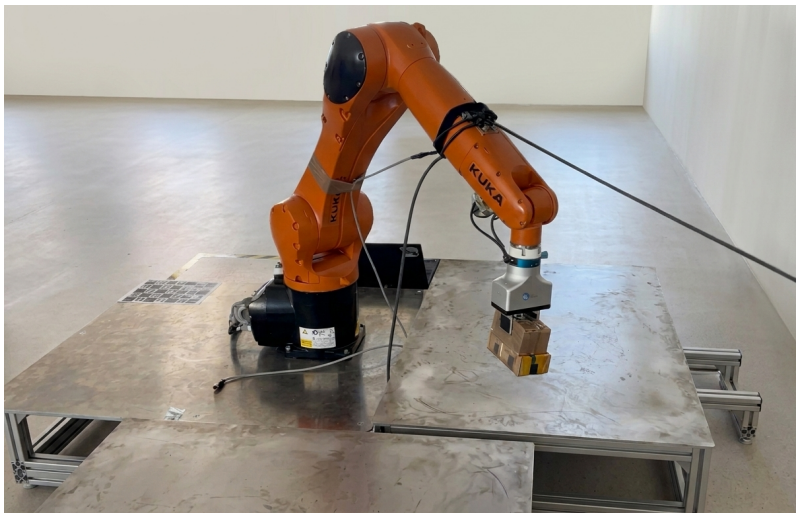


Figure 5: The arm-disturbance fault as physically applied to the KUKA in FactoryWave.

G Details of the FactoryWave dataset

FactoryWave was collected on two one-armed platforms: the **Universal Robots UR3** (cobot; OnRobot Screwdriver and two-finger gripper variants) and the **KUKA KR10** (industrial arm). Each episode corresponds to one complete task cycle; episodes are recorded under three conditions (normal, fault, and counterfactual), described below. Because state interpretation rather than policy quality is the object of study, motion is generated by a deterministic scripted controller. Every sample carries extensive labeling metadata, and the full native-rate sensor stream is logged without online feature selection. Table 8 gives the breakdown of the 8,983 episodes across robots, tasks, and conditions.

Table 8: Distribution of FactoryWave episodes by robot, task, and condition. Counterfactual entries count CF groups (one baseline plus one retained fault run per group), as each one represents data of one studied scenario.

Robot	Task	Normal	Fault	CF groups	Total
UR3	Pick-and-Place	1053	3238	109	4400
UR3	Screwing	230	850	40	1120
UR3	Peg-in-Hole	322	1200	100	1622
KUKA KR10	Pick-and-Place	578	1221	42	1841
Total		**2183**	**6509**	**291**	**8983**

G.1 Tasks and phase structure

Three industrial tasks are represented in the dataset: Pick-and-Place, Screwing, and Peg-in-Hole. All three are executed on both robots, using matched scripted controllers and task layouts so that cross-embodiment comparisons hold the task fixed. We structure each task as a deterministic sequence

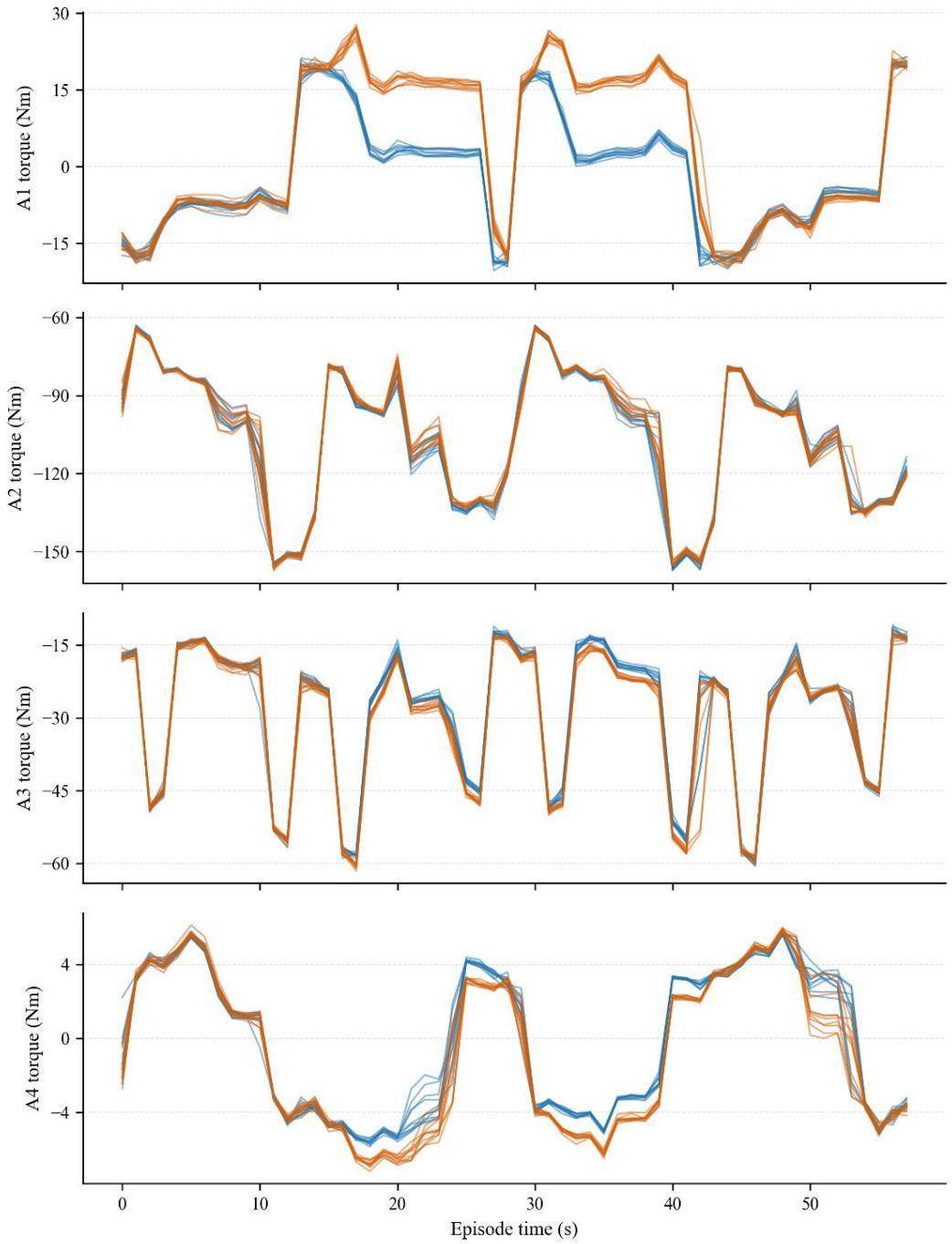


Figure 6: Sensor data distribution across multiple healthy KUKA episodes (blue) versus multiple faulty episodes with an arm-disturbance fault (orange) in FactoryWave. The visible shift between the two groups confirms that templates with small noise-substitution Δ are still solvable from the released signal; the magnitude of the shift varies across joints because the disturbance angle loads them unequally.

of phases (Tables 9, 10, 11); the phase index is written as a discrete label at every timestep, providing an interpretable segmentation that downstream models can condition on or that can be used to localize questions to a specific stage of the cycle. Within this fixed phase skeleton, we randomize as much of the episode as the task permits so that a question template that targets a given phase does not accidentally learn a fixed pose, timing, or trajectory. Pick and place locations are sampled uniformly from predefined workspace regions; phase-specific joint velocities, accelerations, and end-effector (EEF) rotations are drawn from bounded per-phase ranges; and approach angles to contact events are perturbed within the tolerances of the controller. This forces the Q&A pairs generated from FactoryWave to reflect phase-level reasoning rather than memorisation of a single scripted trajectory, which in turn makes them generalizable to a wide variety of trajectories and variation found in real recordings.

Table 9: Task phases for Pick-and-Place. EEF refers to End-Effector

Phase index	Phase name	Description
0	`above_pick`	EEF moves to a waypoint directly above the object
1	`descend`	EEF lowers toward the object
2	`settle`	Brief hold to let physics settle before grasping
3	`close`	Gripper closes around the object
4	`lift`	EEF lifts the grasped object off the surface
5	`move_xy`	Lateral transfer toward the bin
6	`lower`	EEF lowers into the bin
7	`open`	Gripper opens, object released
8	`retract`	EEF moves away from the bin
9	`return`	EEF returns to home configuration

Table 10: Task phases for Screwing.

Phase index	Phase name	Description
0	`approach`	EEF moves to a pre-contact waypoint above the fastener
1	`descend`	EEF lowers to engage the fastener
2	`screw`	Continuous downward force + wrist rotation to thread
3	`disengage`	Gripper opens / tool lifts off
4	`retract`	EEF returns to a safe height
5	`descend`	EEF lowers to engage the fastener (loosening pass)
6	`loosen`	Loosen the screw
7	`engage`	Gripper closes on the retrieved screw
8	`retract`	EEF returns to home position

Table 11: Task phases for Peg-in-Hole.

Phase index	Phase name	Description
0	`above_hole`	EEF moves above the hole and aligns
1	`insert`	EEF pushes peg downward into the hole
2	`disengage`	Gripper opens
3	`above_hole`	EEF rises back to the waypoint above the hole
4	`retract`	EEF returns to home position
5	`above_object`	EEF moves above the object and aligns with it
6	`engage`	Gripper closes
7	`above_object`	EEF rises back to the waypoint above the object
8	`retract`	EEF returns to home position

G.2 Episode metadata

Alongside the sensor stream, every episode carries a fixed set of metadata fields describing the robot, task, condition, payload, and end-effector configuration (Table 12). These fields are constant within an episode and are used both as filters during question generation (e.g. to restrict a template to a specific payload regime) and as provenance in the released Q&A pairs. This includes fault-specific fields (such as injection timestamp or physical offset).

Table 12: Default per-episode metadata fields.

Field	Description
`episode_id`	Unique identifier for the episode
`robot_model`	Robot model (UR3, KUKA KR10)
`task`	Task (pick-and-place, screwing, peg-in-hole)
`condition`	Episode condition (normal, fault, counterfactual)
`fault_id`	Fault index for fault episodes; null otherwise
`weight_of_box`	Mass of the transported object in kg (pick-and-place only)
`shape_of_box`	Shape/dimensions of the transported object (pick-and-place only)
`position_of_box`	Initial XYZ position of the object on the table, where measurable
`gripper_model`	Gripper model and type (vacuum, two-finger, screwdriver)
`payload_mass_configured`	Payload mass configured at the controller (kg)
`payload_cog_configured`	Payload CoG offset configured at the controller (x, y, z in mm)
`tcp_offset_configured`	TCP position offset configured at the controller (x, y, z in mm)
`tcp_orientation_configured`	TCP orientation configured at the controller (rx, ry, rz in radians)

G.3 Fault taxonomy

FactoryWave includes 27 distinct fault types (Table 13). Each fault is paired with a plain-language description of the underlying physical or control-level mechanism and the injection procedure used to induce it during data collection. The last three columns indicate which tasks include the fault in the collected dataset; faults that apply to all three tasks (e.g. collisions with a shared workspace object, misconfigurations of the shared controller) are recorded under each applicable task with task-specific signal imprints.

Table 13: Fault catalogue. Columns *PP*, *Scr*, *PiH* indicate whether the fault is included in the Pick-and-Place, Screwing, and Peg-in-Hole recordings.

Fault	Explanation	Injection	PP	Scr	PiH
Damaged screw thread	Screw thread physically damaged, preventing proper engagement during tightening.	Pre-damaged the screw thread with sandpaper.	×	✓	×
Missing screw	Tightening is attempted with no screw present.	Removed the screw from screwdriver before the cycle.	×	✓	×
Damaged plate thread	The threaded hole in the plate is damaged, preventing engagement.	Pre-damaged the plate hole with a metal screwdriver.	×	✓	×
Loosening phase	A counterclockwise loosening rotation replaces tightening; a normal phase with a distinct signal. It is counted here as an anomaly to match the fault schema of the AURSAD dataset [1].	Reversed the rotation direction in the controller program to execute a counterclockwise loosening pass in place of tightening.	×	✓	×
Gripper activation failure	The vacuum gripper fails to activate and never picks up the box.	Disabled the gripper activation command in the robot program so the pick cycle completes without a grasp.	✓	×	×
Gripper release during motion	The gripper releases the payload mid-trajectory, causing an abrupt payload loss.	Triggered a programmatic gripper-release command at a scripted timestep mid-trajectory.	✓	×	×
Additional axis payload	A dead weight attached to one link increases inertia and gravity loading on all joints.	Bolted calibrated weights of varying mass to one robot link.	✓	✓	×

Table 13 (cont.)

Fault	Explanation	Injection	PP	Scr	PiH
Collision with foam object	Contact with a soft foam block produces a brief TCP force spike without a protective stop.	Placed a foam cube directly in the programmed TCP trajectory.	✓	✓	✓
Unexpected payload weight	The transported box has a weight that deviates from the nominal payload configuration.	Swapped the nominal box for a known heavier or lighter one. This is done in a counterfactual context, ie. switching weights and asking about alternate weight scenario.	✓	×	×
Invalid gripping position	Timing error: the gripper closes after the arm has begun lifting, gripping the object off-center.	Inserted a brief delay in the controller program so gripper closure lags the lift motion.	✓	×	×
Unstable mounting platform	Base instability introduces low-frequency vibrations into the arm.	Placed 8 layers of towels under the base plate.	✓	×	×
Joint position limit violation	A joint moves outside its configured soft or hard position limits, triggering a safety stop.	Random waypoint slightly beyond the configured soft joint limit.	✓	×	×
TCP frame misconfiguration	TCP frame or mounting orientation misconfigured at the controller; gravity torques deviate.	Entered an incorrect TCP offset or mounting angle in the controller installation settings.	✓	✓	✓
Payload weight misconfiguration	Payload mass misconfigured while a tool or workpiece is physically attached.	Left the configured payload weight at multiple wrong mass configurations while a task-dependent gripper remained mounted.	✓	✓	×
External arm disturbance	A continuous external force pulls or pushes the TCP during motion.	Anchored an elastic rope between the bench frame and the tool flange.	✓	✓	✓
Mild payload CoG misconfiguration	Payload CoG specified at an incorrect offset from the tool flange; gravity compensation is wrong.	Entered a CoG offset in the payload configuration that differs from the physical tool CoG.	✓	✓	✓
Collision with hanging cable	A loose cable drags along a robot link or the tool flange during motion.	Suspended a loose cable across the trajectory at arm height.	✓	✓	✓
Collision with cardboard object	A cardboard carton in the trajectory provides moderate resistance; may trigger a protective stop.	Placed a free-standing or braced cardboard box in the programmed trajectory.	✓	✓	✓
Collision with rigid object	A rigid plastic block in the TCP path triggers an immediate protective stop.	Clamped a rigid plastic block in the programmed TCP path. Unlike the other collisions, this fault consistently causes safety stops.	✓	✓	✓
Peg insertion misalignment	Peg approaches the hole at an angular or lateral offset and jams against the rim.	Added a recorded lateral offset at the insertion approach waypoint.	×	×	✓

Table 13 (cont.)

Fault	Explanation	Injection	PP	Scr	PiH
Hole obstruction	Foreign object or debris in the hole prevents full insertion.	Placed a small object (metal screw) inside the hole before insertion.	×	×	✓
Incorrect insertion depth	Insertion terminates at an incorrect Z depth due to a misconfigured waypoint.	Shifted the insertion endpoint waypoint Z by recorded offset from nominal.	×	×	✓
Peg surface contamination	Contamination or roughness on the peg raises insertion friction and produces stick-slip.	Applied dry chalk powder to the peg surface.	×	×	✓
Fixture displacement	Insertion fixture shifted laterally, breaking the match between programmed approach and actual hole.	Physically shifted the hole fixture by recorded offset without updating the robot program.	×	×	✓
Self-collision	Arm configuration causes a link to contact the robot body or mounting fixture, triggering a protective stop.	Random waypoint that forces a self-colliding configuration or offset the mounting fixture into the trajectory.	✓	×	×
Missing box	Box absent from the pick position; the gripper closes on empty air.	Removed the box from the pick position before the episode started.	✓	×	×
Missing peg	Arm descends into the hole empty-handed; insertion produces no contact force.	Removed the peg from the gripper before the episode started.	×	×	✓

G.4 Counterfactual episodes

Counterfactual pairs are constructed for the subset of faults in Table 13 whose injection moment can be controlled at a specific timestep rather than being present throughout the episode. For each such fault, the initial conditions of the physical setup (object identity and pose, payload configuration, controller settings) are held as constant as the platform allows, one baseline execution is recorded without any injection, and several fault executions are then recorded under the same initial conditions with the fault applied at a shared sampled timestep. Because the physical system is not perfectly reproducible, the pre-injection sensor trajectories of the baseline and the fault candidates diverge slightly; we retain, as the counterfactual partner of the baseline, the fault candidate whose pre-injection window is closest to the baseline under the signature kernel MMD, and discard the others. This yields an approximate counterfactual pair in which the early history is as matched as the real platform permits, while the post-injection divergence isolates the effect of the intervention. The approximation error introduced by this procedure, relative to the exact counterfactuals available in simulation, motivates the sim2real analysis in Section J.

G.5 Signal schema

The UR3 is recorded at 125 Hz, yielding 136 per-sample signals grouped by semantic role (Table 14): controller *setpoints*, actual *effort / feedback* readings from joints and the TCP, *context / health* signals covering joint temperatures, voltages, and internal controller state, a small block of controller *status / mode* indicators, and digital/analog *I/O* lines. A handful of episode-level provenance fields (episode identifier, robot type, task, fault label, payload mass, recording date) are appended per episode but are constant within an episode.

The KUKA KR10 is controlled through the KRC4 controller. We stream 53 per-sample signals at 83 Hz, grouped by semantic role in Table 15 using the same partition as the UR3 schema. A handful of signals that the UR3 exposes natively are unavailable on KSS 8.3, specifically joint velocities, commanded TCP pose, joint voltage, and joint-level control outputs; TCP force/torque

Table 14: UR3 signal groups recorded at 125 Hz (excluding episode-level provenance columns).

Group	# signals	Contents
Setpoint (intent)	42	Target joint position, velocity, acceleration, current, torque; target TCP pose and speed.
Effort / feedback	48	Actual joint position, velocity, current, current-as-torque, joint control output; actual TCP pose, speed, and force/torque.
Context / health	33	Joint temperatures, joint voltages, joint modes, tool accelerometer, raw F/T wrench, main/robot voltage and current, momentum, speed scaling, execution time.
Status / mode	7	Timestamp, robot mode, robot status, safety mode, safety status bits, runtime state, configured payload mass.
I/O	6	Digital inputs/outputs, two analog inputs, two analog outputs.
Total	**136**	

is also absent because no force sensor is fitted. To recover tool-side feedback, we mount a 9-DoF inertial measurement unit on the gripper and log its triaxial linear acceleration (`acc_x/y/z`), angular rate (`gyro_x/y/z`), and orientation (`angle_x/y/z`) at the controller's 83 Hz cadence.

Table 15: KUKA KR10 signal groups recorded at 83 Hz via RSI/KRL (excluding episode-level provenance columns).

Group	# signals	Contents
Setpoint (intent)	6	Commanded joint positions (degrees).
Effort / feedback	24	Actual joint positions (degrees), actual TCP pose (mm / degrees), per-axis motor current (% of max), and per-axis motor torque (Nm).
Context / health	11	Per-axis motor temperature (Kelvin), Cartesian acceleration on X, Y, Z and absolute magnitude (m/s^2), and speed override (%).
Tool IMU	9	Gripper-mounted 9-DoF IMU: triaxial linear acceleration `acc_x/y/z` (m/s^2), angular rate `gyro_x/y/z` (rad/s), and orientation `angle_x/y/z` (degrees).
Status / mode	1	Controller process state (FREE / ACTIVE / STOP / END).
I/O	2	Digital inputs bitmask and digital outputs bitmask.
Total	**53**	

H Time-series foundation model baseline (Chronos-Bolt)

To check whether the LLM panel is the right comparison set on a benchmark whose motivation references time-series models, we evaluate CHRONOS-BOLT [17] ($\sim$200 M parameters), a pretrained probabilistic time-series forecaster, as a non-LLM baseline in the same zero-shot setting (no fine-tuning, prompting, or context-window engineering on either side).

Template applicability. A pure forecaster has neither a language interface nor a notion of options, ranking, or counterfactual conditioning. Of the 21 active question templates in FactoryBench, only three reduce to a well-posed forecasting problem: L1.7 (scalar forecast), L2.4 (scalar forecast under a known anomaly), and L2.5 (six-joint tensor forecast under a known anomaly). The remaining 18 templates (MCQ classification, ranking, counterfactual conditioning, free-form root-cause and protocol writing) are structurally outside the scope of any TSFM that lacks a language head. **This 14 % template coverage is itself a finding**: any TSFM, regardless of size or pretraining quality, can address only the forecasting slice of FactoryBench, and the other 86 % of the benchmark measures capabilities that no current TSFM provides.

Setup. For each forecasting item we form a univariate context per target channel, request the median quantile at the requested horizon, and score under the same chance-corrected piecewise-margin rule used in the main paper (Appendix A; $E = 1/4$). For the tensor template (L2.5) we run six independent per-joint forecasts and average the per-channel scores. Chronos-Bolt has no multivariate or text-conditioned mode, so cross-channel correlations and the anomaly named in the question are unavailable to it.

Results. On the three forecasting templates of the test split (Table 16, $n = 469$), Chronos-Bolt achieves chance-corrected accuracy of 51.5%, 46.9%, and 49.7% on L1.7, L2.4, and L2.5 respectively. As a reference point, the strongest LLM in the panel reaches 46.8% on the L1 level aggregate and 47.1% on the L2 aggregate (Section 5.1); a strict per-template re-score of the LLM panel is left to follow-up work.

Table 16: CHRONOS-BOLT zero-shot performance on the three forecasting templates of FactoryBench (test split, $n = 469$). Confidence intervals are 95 % percentile intervals from 1000 bootstrap resamples. Chance-corrected (CC) scores apply $\max(0, (s - 1/4)/(1 - 1/4))$.

Template	n	Raw mean	Raw 95% CI	CC mean	CC 95% CI
L1.7 (state, scalar)	334	0.636	[0.588, 0.681]	0.515	[0.451, 0.575]
L2.4 (intervention, scalar)	74	0.601	[0.493, 0.710]	0.469	[0.324, 0.614]
L2.5 (intervention, tensor[6])	61	0.623	[0.537, 0.708]	0.497	[0.383, 0.611]

Score distribution and signal coverage. Per-item scores are strongly bimodal: items tend to land at 1.0 or 0.0, with a smaller mass at the half-credit band, characteristic of a margin-thresholded scorer. Aggregating items by target-signal family shows roughly uniform capability across the four families (positions, speeds, torques/efforts, contact forces) within bootstrap noise, indicating the baseline does not selectively succeed on smooth signals and fail on noisy ones. Score declines monotonically with forecast horizon, as expected for a probabilistic forecaster.

Fairness of comparison. The comparison is apples-to-apples on item identity, scoring rule, chance correction, and zero-shot setting. It is *not* apples-to-apples on (i) input representation: LLMs see acronymized text-encoded values, alongside the question, robot description, and option set, while Chronos-Bolt sees raw univariate float arrays only; and (ii) question awareness: on L2.4 and L2.5 the LLM is told which anomaly is present, while Chronos-Bolt cannot use that information. A multivariate or text-conditioned TSFM (e.g. Moirai-MoE, Time-MoE) might exploit these signals; we do not, to keep the baseline architecturally minimal. We therefore report Chronos-Bolt numbers as template-specific rather than as level aggregates.

Interpretation. Two messages emerge. **First, on the templates where forecasting is the right tool, a 200 M-parameter pretrained forecaster is competitive with frontier LLMs**, consistent with the main paper's observation that LLMs struggle with precise numerical state extraction at L1. The LLM panel is not the only, nor the most viable model class on the forecasting slice. This opens the door to tool-assisted LLM agents that delegate forecasting subtasks to a specialised TSFM and reserve the LLM for the linguistic, classification, and decision-making templates where it has the structural advantage. **Second, the remaining 18 of 21 FactoryBench templates (classification, ranking, counterfactual reasoning, free-form troubleshooting and optimization) are structurally outside any current TSFM's scope.** Together these reinforce the paper's central claim: the gap between current models and operational machine understanding is not closed by solely scaling LLMs or deploying purpose-built forecasters.

I Example question-answer pairs and question templates

Table 17 presents seventeen representative Q&A pairs spanning every causal level and every answer format the generator produces (single-select MCQ, multi-select MCQ, tensor, ranking, free-form). For readability we omit the underlying sensor context (tens to hundreds of time-series rows), truncate verbose boilerplate such as "Answer only with . . .", and abbreviate long option strings.

Table 17: Representative Q&A pairs from FactoryBench spanning all four causal levels and every answer format.

Level	Answer format	Prompt, options, and reference answer
L1	Tensor	**Q:** The robot is performing a screwing task. We want to isolate the screwing phase in the robot's time series. Assuming a fixed window length of 10 timesteps, at which timestamp should the window begin? **A:** `14` (accepted within ± 3)
L1	Single-select MCQ	**Q:** What changed between the two instances of robotic time series data? **Options:** **a.** different anomalous states (exactly one anomalous, or both but distinct) **b.** different tasks **c.** different robots **d.** same task, different phases **A:** `a`
L1	Single-select MCQ	**Q:** What robot does this sensor data originate from? **Options:** **a.** Agile Robots Yu 5 Industrial **b.** KUKA KR 10 R1100-2 **c.** Universal Robots UR3 **A:** `c`
L1	Tensor	**Q:** Given the sensor stream below, what is the expected value of the position of joint 2 at T+6 ms? **A:** `102.7777`
L2	Tensor	**Q:** The sensor stream below is from a robot exhibiting a collision with a cardboard object. What is the expected value of the velocity of joint 3 at T+68 ms? **A:** `-0.272044` (accepted within ± 2.42)
L2	Tensor	**Q:** The sensor stream below is from a robot exhibiting a collision with a cardboard object. What are the expected values of joint positions at T+506 ms? **A:** `[17.884, -115.897, 121.049, -95.595, -90.258, 293.860]`
L2	Tensor	**Q:** Knowing that the robot suffers from a hole obstruction, we want to isolate the "rise back above the object" phase. Assuming a fixed window length of 16 timesteps, at which timestamp should the window begin? **A:** `202` (accepted within ± 3)
L2	Single-select MCQ	**Q:** Given the sensor data, determine what anomaly is present. **Options:** **a.** sustained command/measurement deviation **b.** no anomaly **c.** measured joint speeds drop abruptly with a TCP force spike (typical collision signature) **d.** isolated unrealistic force-sensor spike inconsistent with motion **A:** `c`
L3	Ranking	**Q:** Given the baseline stream and the counterfactual scenario where a collision foam object occurs at timestep 7366 ms, rank the four labeled signal segments in the order they would appear as the event manifests. **A:** `abdc`
L3	Multi-select MCQ	**Q:** Given the counterfactual scenario where a collision hanging cable occurs at timestep 8363 ms, select all statements that would apply. **Options:** **a.** motor current rises $\geq 33\%$ while speed stays $\leq 10\%$ of baseline (stall-like) **b.** contact-force spike $\geq 36\%$ above baseline **c.** tracking-error rise $\geq 34\%$ above pre-event mean **d.** at least one joint speed drops sharply ($\geq 31\%$) **A:** `FFFT`

continued on next page

Table 17 continued from previous page

Level	Answer format	Prompt, options, and reference answer
L3	Tensor	**Q:** In the counterfactual scenario where a collision foam object occurs at timestep 7659 ms, what would be the expected value of the position of joint 2 at T+100 ms? **A:** `72.144939` (accepted within ± 2.77)
L4	Free form	**Q:** Given the sensor stream below, does the machine show signs of anomalous behavior? If yes, identify the most likely root cause and describe the steps you would take to fix it. **A:** No anomalous behavior detected; the machine is operating normally; no remediation is required.
L4	Free form	**Q:** Given the sensor stream below, does the machine show signs of anomalous behavior? If yes, identify the most likely root cause and describe the steps you would take to fix it. **A:** **a.** Inspect the workspace and remove any soft obstructions from the programmed trajectory. **b.** If a protective stop occurred, acknowledge it in the robot controller and verify arm posture before resuming. **c.** Reduce speed if repeated false-positive collision detections occur.
L4	Free form	**Q:** Given the sensor stream below, does the machine show signs of anomalous behavior? If yes, identify the most likely root cause and describe the steps you would take to fix it. **A:** **a.** Stop the robot and update the payload configuration to include the additional axis weight. **b.** Set the correct CoG offset for the modified arm in the installation settings. **c.** Reduce motion accelerations and speeds to account for increased effective inertia. **d.** Run a slow-speed test cycle to verify dynamic loads stay within rated limits before resuming.
L4	Free form	**Q:** An engineer wants to increase the effectiveness and accuracy of this machine. Based on the sensor stream below, what operational changes or parameter adjustments could help achieve this? **A:** The payload mass is set to 0.0 kg but the actual payload weighs 1.5 kg. Update the payload mass in the installation settings to 1.5 kg.
L4	Free form	**Q:** An engineer wants to increase the effectiveness and accuracy of this machine. Based on the sensor stream below, what operational changes or parameter adjustments could help achieve this? **A:** The TCP offset is misconfigured at [0, 0.3, 55] instead of the correct [0, 0, 55]. Update the TCP position offset in the installation settings to [0, 0, 55].
L4	Free form	**Q:** An engineer wants to increase the effectiveness and accuracy of this machine. Based on the sensor stream below, what operational changes or parameter adjustments could help achieve this? **A:** The payload center of gravity is set to [25, 0, 20] but the correct value is [0, 0, 20]. Update the CoG offset in the installation settings to [0, 0, 20].

J Simulation setup and sim-to-real gap analysis

We run sim2real rollouts in Isaac Sim using the built-in UR3 asset, with phase-consistent waypoint replay of the recorded `target_joint_*` signals. The simulated task is UR3 pick-and-place, following the 10-phase structure of Table 9. Payload settings are matched episode-wise to the FactoryWave chronology.

Each real episode is exported, converted to a per-episode simulation config, and replayed in Isaac. Simulated outputs preserve the FactoryWave field schema (`joint_*`, `tcp_*`, `task_phase`, etc.) and are paired with their real counterpart by episode ID. Gap metrics are computed phase-wise on aligned trajectories: joint RMSE (deg), TCP position RMSE (mm), TCP rotation error (mrad), and Wasserstein-1 on `joint_current_*` (A).

Across 1,155 paired episodes, pooled per-episode metrics are: joint RMSE mean/median $= 3.65/2.83$ deg, TCP position RMSE $= 13.16/13.26$ mm, and W1 effort mean $= 0.83/0.81$ A. TCP rotation error remains broad (median 2605 mrad), indicating orientation mismatch is the least stable component. Translational kinematics and effort-proxy alignment are substantially more consistent than rotational alignment.

K Causal schema (SCE): full details

This appendix gives the long-form definition of the Setpoint–Context–Effort + Feedback (SCE) schema used to organize every time-series channel in FactoryBench, together with the residual-based fault definition that the schema enables. The motivation is summarized in Section 3.2; here we cover the per-group channel breakdown, the residual formalism, and the per-robot calibration that makes faults computable rather than imputed.

The schema groups every recorded channel into one of three causal roles. **Setpoint** captures the controller's commanded target: per-axis position, velocity, and acceleration setpoints emitted by the trajectory generator. **Context** captures physical and configured conditions affecting behavior, split into *static* context (episode-level metadata such as payload mass, payload center of gravity, TCP offset, gripper model, and box / peg geometry) and *dynamic* context (per-sample channels such as joint temperatures, supply voltages, controller mode, and safety mode). **Effort + Feedback** captures the machine's measured response: motor currents and torques (effort), measured joint positions and TCP pose (feedback), tool accelerometer, and acoustic emission where available. The exact per-robot channel mapping appears in the signal-group tables in this appendix (UR3 and KUKA KR10).

This split enables a single operational definition of a fault that applies across machines and tasks. Under healthy operation, the measured Effort + Feedback response $\mathbf{y}_t$ is by definition a function of the Setpoint command $\mathbf{S}$ and the Context $\mathbf{C}_t$:

$$\mathbf{y}_t = f(\mathbf{S}, \mathbf{C}_t),$$

where f represents the nominal plant dynamics (inverse-dynamics-plus-controller transfer) calibrated per robot. Faults manifest as structured residuals from this expected baseline: $\mathbf{y}_t - f(\mathbf{S}, \mathbf{C}_t)$ forms a clear, fault-specific spatiotemporal pattern. Because every episode supplies the paired $(\mathbf{S}, \mathbf{C}_t, \mathbf{y}_t)$ streams, this residual is computable directly from the data, giving ground truth that does not have to be hand-imputed. Figure 7 summarises the causal relationships between the three groups.

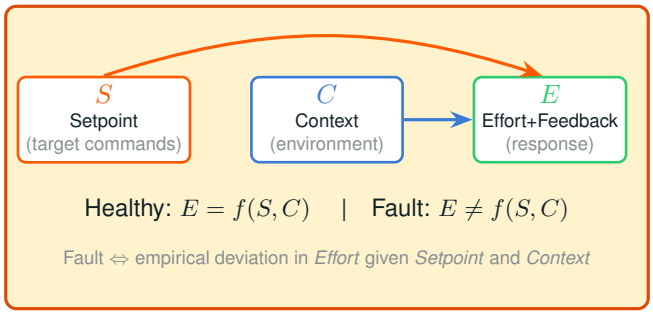


Figure 7: Physical causal schema. Setpoint commands S and contextual variables C jointly determine the measured Effort + Feedback response E under healthy operation. Faults manifest as residuals $E - f(S, C)$ from this nominal mapping.

L License and availability

FactoryBench and FactoryWave are released under the MIT License and are freely available for academic and commercial use. Both the episode data and the benchmark artefacts (question templates, paraphrase banks, LLM-as-judge prompts, generator source code, and the full Q&A dataset) are distributed via the public Hugging Face repository `FactoryBench/FactoryBench`. To guard against test-set contamination in future model releases, we hold back a private 15% subset of the Q&A pairs at the *episode* level (so all questions grounded in any held-out episode stay private across all four

levels) and keep its raw episode data, gold answers, and rubric annotations off the public release; this private slice is used internally for contamination checks and may also be used to score future evaluees on request. The remaining 85% is fully public, with a train/validation/test split included in the release so that evaluees can reproduce the exact partition used in this paper.

We provide a versioned release track (e.g., `v1.0`, `v1.1`) so that reported numbers always refer to a fixed benchmark state; corrections to labels or templates are published as minor-version updates with a public changelog, and the exact version used in any evaluation is stamped into every result file. Bug reports and label-correction requests are tracked on a public repository (link provided upon acceptance).