# When Simulation Lies: A Sim-to-Real Benchmark and Domain-Randomized RL Recipe for Tool-Use Agents

**Xiaolin Zhou** [1] **Aojie Yuan** [2] **Zheng Luo** [2] **Zipeng Ling** [3] **Xixiao Pan** [2] **Yicheng Gao** [2] **Haiyue Zhang** [2] **Jiate Li** [2] **Shuli Jiang** [4] **Prince Zizhuang Wang** [4] **Zixuan Zhu** [2] **Jinbo Liu** [1] **Ryan A. Rossi** [5] **Hua Wei** [1] **Xiyang Hu** [1]

## Abstract

Tool-use language agents are evaluated on benchmarks that assume clean inputs, unambiguous tool registries, and reliable APIs. Real deployments violate all these assumptions: user typos propagate into hallucinated tool names, a misconfigured request timeout can stall an agent indefinitely, and duplicate tool names across servers can freeze an SDK. We study these failures as a sim-to-real gap in the tool-use *partially observable Markov decision process (POMDP)*, where deployment noise enters through the observation, action space, reward-relevant metadata, or transition dynamics. We introduce **RobustBench-TC**, a benchmark with 22 perturbation types organized by these four POMDP components, each grounded in a verified GitHub issue or documented tool-calling failure. Across 21 models from 1.5B to 32B parameters (including the closed-source o4-mini), the robustness profile is sharply uneven: observation perturbations reduce accuracy by less than 5%, while reward-relevant and transition perturbations reduce accuracy by roughly 40% and 30%, respectively; scale alone does not close these gaps. We then propose **ToolRL-DR**, a domain-randomization reinforcement learning (RL) recipe that trains a tool-use agent on perturbation-augmented trajectories spanning the three statically encodable POMDP components. On a 3B backbone, ToolRL-DR-Full retains roughly three-quarters of clean accuracy and reaches an aggregate perturbed accuracy comparable to open-source 14B function-calling baselines while substantially narrowing the gap to o4-mini. It closes approximately 27% of the Transition gap despite never seeing transition perturbations in training, suggesting that RL on adversarial static tool-use inputs induces a more persistent retry policy that transfers to unseen runtime failures.

[1]Arizona State University [2]University of Southern California [3]University of Pennsylvania [4]Carnegie Mellon University [5]Adobe Research. Correspondence to: Xiaolin Zhou <xzhou226@asu.edu>, Xiyang Hu <xiyanghu@asu.edu>.

*Failure Modes in Agentic AI Workshop, The $43^{rd}$ International Conference on Machine Learning*, Seoul, South Korea. 2026.
Copyright 2026 by the author(s).

## 1. Introduction

Tool-using language agents are increasingly deployed in production systems: agent frameworks such as LangChain and AutoGen, MCP-based assistants, and retrieval or code-execution back ends for chat products. Yet the benchmarks used to evaluate these agents (BFCL V3 (Patil et al., 2024), API-Bank (Li et al., 2023), ToolAlpaca (Tang et al., 2023), RoTBench (Ye et al., 2024b), and others) still score behavior under clean conditions: user queries are well formed, tool registries are unambiguous, tool descriptions are stable, and tool execution is deterministic. Production traffic violates all of these assumptions. A user typo such as "occcra_information" → "occrra_information" can be copied into a hallucinated tool name and crash a LlamaIndex dispatcher (LlamaIndex contributors, 2023). A LangChain client with default *request_timeout=None* can hang indefinitely on slow tool responses (LangChain contributors, 2025c). Two MCP servers can register tools with the same name and freeze the OpenAI Agents SDK (openai-agents-python contributors, 2025). Clean benchmarks do not expose these failures, so models tuned against them receive little signal about how to handle realistic tool-calling errors.

We study this mismatch as a sim-to-real transfer problem. In robotics, a standard response to a training–deployment distribution gap is domain randomization (Tobin et al., 2017; Peng et al., 2018; Sadeghi & Levine, 2017; Chebotar et al., 2019): train on a broader distribution of environment variations so that deployment-time conditions are less likely to fall outside the training support. Tool use has a similar structure, but the sources of variation differ. Instead of sensor noise, visual textures, or actuator dynamics, tool-use agents face noisy text, ambiguous action spaces, misleading tool metadata, and unreliable tool execution. We formalize these sources through the tool-use partially observable Markov

decision process (POMDP): deployment noise can enter the observation, the action space, the reward-relevant metadata used to choose among tools, or the transition dynamics induced by tool execution.

Existing tool-use robustness work studies important but mostly isolated failure modes, such as function-name perturbations (Ye et al., 2024b), distractor tools (Lu et al., 2025), and query or toolkit modifications (Rabinovich & Anaby-Tavor, 2025). These studies show that tool-use agents are brittle, but they do not provide a unified POMDP-based taxonomy, do not systematically tie perturbation types to production failures, and do not test whether training on perturbed tool-use trajectories can reduce the gap. As a result, it remains unclear which part of the tool-use loop is most fragile, whether scaling alone improves robustness, and whether a data-side training fix can transfer beyond the perturbations seen during training.

In this work, we make three contributions:

- **RobustBench-TC** (§4): a production-grounded benchmark of 199 single-turn samples drawn from five existing tool-use benchmarks, with 22 perturbation types organized by POMDP component (observation, action, reward, transition). Every type is linked to a verified GitHub issue from a major agent framework (LangChain, AutoGen, OpenAI Agents, LlamaIndex, MCP servers, etc.) or to a peer-reviewed study, with an evidence audit released with the data.
- **ToolRL-DR** (§5): a domain-randomization RL training recipe for tool use. ToolRL-DR keeps the public ToolRL training code fixed, but replaces clean training trajectories with perturbation-augmented trajectories sampled from the three statically encodable POMDP components. On a 3B backbone, ToolRL-DR-Full closes approximately 27% of both the Reward and the Transition gap relative to the public ToolRL checkpoint, despite never training on transition perturbations.
- **A live leaderboard** (§6.5): a public submission portal that scores user-supplied predictions against RobustBench-TC, reports per-component robustness, and tracks tool-use robustness over time, following the submission pattern of community leaderboards such as BFCL.

We find that the sim-to-real gap is large and uneven, scaling does not reliably close it, and ToolRL-DR partially transfers to unseen Transition perturbations on a 3B backbone.

## 2. Background

### 2.1. Tool use as a partially observable Markov decision process

We treat a tool-using agent as a partially observable Markov decision process (POMDP) $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \Omega, \mathcal{O} \rangle$. The hidden state $s_t \in \mathcal{S}$ encapsulates the user's underlying intent, the actual implementation of every tool, and the runtime health of every service. The agent does not access $s_t$ directly; instead it receives an observation $o_t \in \Omega$ produced by the observation function $\mathcal{O}(o_t \mid s_t)$, comprising the dialog prompt, the tool registry it can see (names, descriptions, parameter schemas), and the outputs of any tools already executed. An action $a_t \in \mathcal{A}$ is a structured tool invocation, i.e. a pair (tool name, argument map). The transition $\mathcal{P}(s_{t+1} \mid s_t, a_t)$ executes the chosen tool and updates the hidden state; the agent then receives a new observation. The reward $\mathcal{R}(s_t, a_t)$ scores whether the invocation is correct and, in deployed systems, may also depend on cost, latency, or other metadata relevant to tool choice.

The agent samples actions from a history-dependent policy $\pi_\theta(a_t \mid h_t)$ where $h_t = (o_1, a_1, \ldots, o_t)$ is the observation–action history; in our setting the policy is a function-calling LLM that consumes $h_t$ as a prompt. Each POMDP component is a place where simulation and reality can differ: the observation function $\mathcal{O}$ may produce noisy text the agent did not see in training; the action space $\mathcal{A}$ may include distractor tools; the transition $\mathcal{P}$ may produce transient errors not present in the curated benchmark; and the reward-relevant metadata visible inside $o_t$ may be misleading. We use this decomposition to organize our perturbations (§4).

### 2.2. Sim-to-real and domain randomization

Sim-to-real transfer studies policies trained in a source environment, often a simulator, and deployed in a target environment whose conditions differ. In robotics, a standard response to this training–deployment gap is *domain randomization* (Tobin et al., 2017; Sadeghi & Levine, 2017; Peng et al., 2018; Tan et al., 2018): during training, sample environment parameters such as textures, dynamics, masses, friction, or sensor noise from a broad distribution so that the target environment is less likely to fall outside the training support. Adaptive variants go further by updating the randomization distribution *online* during training, either by progressively expanding the range as the agent's competence grows (Akkaya et al., 2019) or by leveraging real-world rollouts to close the simulation-to-reality loop (Chebotar et al., 2019). Further details on randomization axes and transfer settings are provided in (Zhao et al., 2020; Höfer et al., 2021).

We apply the same lens to tool-use agents. The perturbation axes differ from robotics: noisy text replaces sensor noise, transient API errors replace actuator or dynamics mismatch, and distractor tools replace visual or physical clutter. The underlying problem is similar: clean training and benchmark data cover a narrower distribution than the conditions encountered during deployment. Concretely, we organize tool-use perturbations along the four POMDP com-

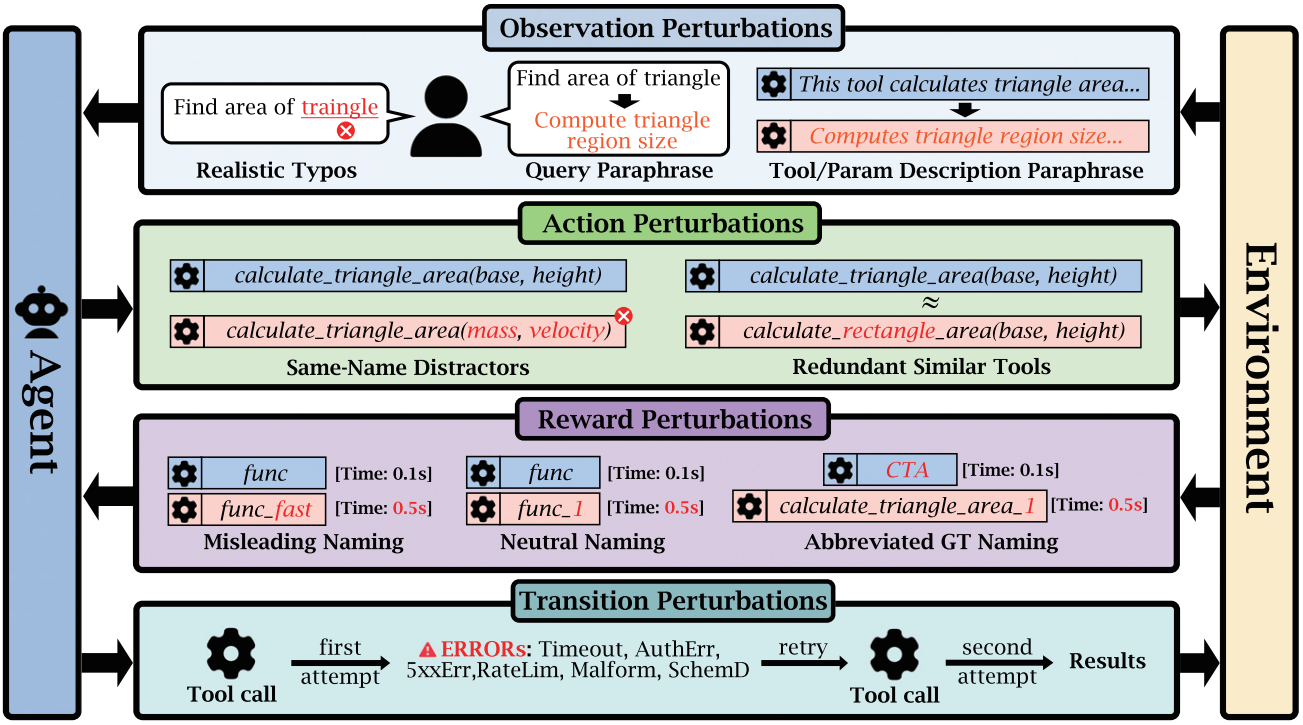



*Figure 1.* The tool-use POMDP and the four perturbation categories that define RobustBench-TC. Information flows between the **Agent** (left) and the **Environment** (right) through four perturbation channels arranged top-to-bottom. **Observation** perturbations corrupt what the agent reads from the environment (typos in the user query, paraphrases of the query, paraphrases of tool or parameter descriptions). **Action** perturbations contaminate the available action space (a same-name distractor that swaps argument names, or a redundant similar tool). **Reward** perturbations corrupt the metadata that disambiguates among similarly-named tools (misleading naming, neutral naming, or an abbreviated ground-truth name). **Transition** perturbations inject a transient runtime error (Timeout, AuthErr, 5xxErr, RateLim, Malform, SchemD) after the model's first tool call, forcing a retry. Type-level details are in Table 1; production grounding for each type is in §4.2.

ponents: **Observation** (noisy text in user queries and tool metadata), **Action** (distractor tools in the action space), **Reward** (misleading metadata that biases tool selection), and **Transition** (transient runtime errors during execution). The full type-level taxonomy and grounding evidence are in §4.1. Section 5 instantiates domain randomization for tool calling by training on perturbation-augmented trajectories rather than only clean tool-use trajectories.

## 3. Related Work

**Tool-use LLM agents** Early tool-use agents combined Chain-of-Thought reasoning (Wei et al., 2022) and ReAct-style observation loops (Yao et al., 2023) with supervised fine-tuning on curated tool-use trajectories (Schick et al., 2023; Qin et al., 2024; Tang et al., 2023; Liu et al., 2024). Recent systems increasingly use reinforcement learning to optimize tool-call behavior. ToolRL (Qian et al., 2025) applies GRPO with structured rewards over synthetic tool-call trajectories. TL-Training (Ye et al., 2024c) combines supervised fine-tuning with token-weighted reinforcement learning. LoopTool (Zhang et al., 2025) uses judgment-guided label refinement to close a data–training loop. MUA-RL (Zhao et al., 2025) trains with simulated users in multi-turn rollouts. We evaluate representatives of these RL-trained tool-use families, together with their base or instruction-tuned counterparts where applicable.

**Tool-use benchmarks** Tool-use benchmarks have expanded across task format, tool diversity, and interaction length. API-Bank (Li et al., 2023) evaluates planning, retrieval, and API invocation. The Berkeley Function-Calling Leaderboard (BFCL) (Patil et al., 2024) evaluates cross-domain function calling at scale. ACEBench (Chen et al., 2025) focuses on multi-turn agent dialogues. $\tau$-bench (Yao et al., 2024) evaluates conversational agents in dual-control environments. ToolAlpaca (Tang et al., 2023), ToolEyes (Ye et al., 2024a), and related datasets increase tool variety and output-format diversity. These benchmarks primarily test clean conditions: queries are well formed, tool descriptions are stable, action spaces are unambiguous, and tool execution is deterministic. RobustBench-TC uses five of these benchmarks as source data and applies a production-grounded perturbation taxonomy to each.

**Robustness perturbations for LLMs and tool use** General LLM robustness work measures sensitivity to typos,

paraphrases, and synonym substitutions (Raj et al., 2022). Tool-use robustness work has studied several important failure modes. RoTBench (Ye et al., 2024b) perturbs function names and parameters. ToolSandbox (Lu et al., 2025) introduces distractor tools and sequencing tests. Rabinovich & Anaby-Tavor (2025) evaluate query rephrasing and toolkit modifications on BFCL, reporting 13–19% degradation, with many errors caused by parameter-value mismatches. Faghih et al. (2025) show that tool-description wording can strongly affect LLM tool preferences. RobustBench-TC differs in three ways: it organizes perturbations by POMDP component, ties perturbation types to verified production failures or documented empirical effects, and pairs the benchmark with a domain-randomized training recipe.

**Sim-to-real beyond robotics** Sim-to-real transfer has been widely studied in robotic control with deep reinforcement learning (Tobin et al., 2017; Peng et al., 2018; Sadeghi & Levine, 2017; Chebotar et al., 2019; Tan et al., 2018; Akkaya et al., 2019), with surveys covering common transfer settings and randomization strategies (Zhao et al., 2020; Höfer et al., 2021). Our work moves this framing to tool-use language agents. Rather than randomizing physical parameters, we randomize text inputs, tool registries, and reward-relevant metadata in tool-use trajectories, then test whether this reduces deployment-style failures (§5).

## 4. RobustBench-TC

We build RobustBench-TC on top of five existing tool-use benchmarks: BFCL V3 single-turn (Patil et al., 2024), API-Bank (Li et al., 2023), RoTBench (Ye et al., 2024b), ToolAlpaca (Tang et al., 2023), and ToolEyes (Ye et al., 2024a). We select these because (i) their tasks are released under permissive licenses for redistribution, (ii) together they span the major tool-call output formats (BFCL bracketed AST, JSON *¡tool_call¿*, ReAct), and (iii) they are the benchmarks used by the published RL-trained tool-use models (Qian et al., 2025; Zhang et al., 2025; Zhao et al., 2025; Ye et al., 2024c) we evaluate against. Per-source details are in Appendix A.

### 4.1. Perturbation taxonomy

We organize perturbations by the four elements of the POMDP (§2.1). Within each category we instantiate concrete types targeting distinct failure modes documented in production. Figure 1 sketches the four categories with one canonical example per type; the full taxonomy is summarized in Table 1. We use short identifiers in figures and tables; the mapping to the longer code-level names is in Appendix D.

### 4.2. Production grounding

A perturbation type is worth measuring only if it represents a failure that occurs in deployed systems. We tie each of the 22 types to one or more verified GitHub issues from major agent frameworks (LangChain, AutoGen, OpenAI Agents, LlamaIndex, MCP servers) or to peer-reviewed studies. Every cited issue was re-fetched manually to confirm both URL existence and content match; we collected sources only from tool-calling / agent / MCP / function-calling repositories. Representative examples include: a user typo propagates into a hallucinated tool name; a paraphrased query routes to the wrong tool; two MCP servers register the same tool name and the SDK hangs; LangChain's default *request_timeout=None* hangs agents forever. The full type→canonical issue map (Table 5) and audit log are in Appendix E.

### 4.3. Benchmark construction

**Offline perturbations.** Observation, Action, and Reward perturbations modify static benchmark data. LLM-generated types (Typo, QueryPara, ToolPara, ParamPara, RedunTool, and the description rewriting in MisDesc / TimeDesc) use *gpt-5-mini* (OpenAI, 2025) with the prompts in Appendix F. The `realistic_typos` type is an exception: stronger aligned models tend to silently auto-correct or refuse to inject typos, so we instead use *gpt-4o-mini* (OpenAI, 2024), which is more willing to produce lexically-noisy text. Rule-based types (Dup-NoDesc through Dup-SwapDP, and the six reward-variant suffixings) are deterministic. Semantic equivalence of LLM-generated outputs is in Appendix H.

**Online perturbations.** Transition perturbations modify execution dynamics rather than static data. When the model emits its first tool call, the harness intercepts it and returns a transient-error message (one of timeout, HTTP 429, HTTP 401/403, HTTP 5xx, malformed JSON, or schema drift; strings in Appendix G) in place of the actual tool result, giving the model a chance to recover on its next turn. The scorer evaluates the recovery turn: a model that retries with the correct tool earns full credit, while one that gives up or repeats a wrong call earns zero. We inject the transition perturbation at the first tool call so that every model is tested at the same trajectory step, which isolates the per-error-type effect from confounds introduced by the position at which the error occurs. We use a 100% injection rate per sample for the same reason.

The clean baseline contains 199 samples drawn from the five source benchmarks. Applying the 22 perturbation types to whichever sources support each yields 3,522 perturbation samples (ToolEyes is the only exception: it lacks a single ground-truth tool with a distractor structure and therefore skips the 6 Action and 6 Reward types, while still contribut-

*Table 1.* Perturbation taxonomy by POMDP component. The *display name* is the abbreviated identifier used in figures and tables throughout the paper; the corresponding code-level name (used in our released datasets) is in Appendix D. Generation method indicates how each perturbation is produced from a clean source sample.

| Component | Display name | What is modified | Method |
|---|---|---|---|
| Observation | Typo | user query ( character-level keyboard noise ) | LLM |
| | QueryPara | user query ( semantic rephrasing ) | LLM |
| | ToolPara | tool description | LLM |
| | ParamPara | parameter description | LLM |
| Action | Dup-NoDesc | inject distractor sharing GT name; no desc , no params | rule |
| | Dup-Desc | same name; correct description , no params | rule |
| | Dup-WrongP | same name; no description , wrong params | rule |
| | Dup-DescWP | same name; correct description , wrong params | rule |
| | Dup-SwapDP | same name; swapped description , wrong params | rule |
| | RedunTool | inject functionally similar but incorrect distractor | LLM |
| Reward | MisDesc | misleading description on GT; *_Budget/_Fast* suffix on distractor | rule |
| | TimeDesc | response-time annotation ; same naming pattern | rule |
| | MisDesc-N | MisDesc + neutral suffix on distractor ( *_1* ) | rule |
| | TimeDesc-N | TimeDesc + neutral suffix | rule |
| | MisDesc-Abbr | MisDesc + abbreviated GT name | rule |
| | TimeDesc-Abbr | TimeDesc + abbreviated GT name | rule |
| Transition | Timeout | first tool call returns " Tool execution timed out " | runtime |
| | RateLim | first call returns HTTP 429 | runtime |
| | AuthErr | first call returns HTTP 401/403 | runtime |
| | 5xxErr | first call returns HTTP 5xx | runtime |
| | Malform | first call returns malformed JSON | runtime |
| | SchemaD | first call returns " parameter X no longer valid " | runtime |

ing Observation and Transition). Overall, the whole evaluation set comprises 3,721 samples, and the full breakdown is in Appendix A.

## 5. ToolRL-DR: Domain Randomization for Tool-Use RL

### 5.1. Motivation

Tool-use agents fail under realistic perturbations they did not see at training time (§4.2). Motivated by domain randomization in robotics (Tobin et al., 2017; Peng et al., 2018; Sadeghi & Levine, 2017), we apply the same principle to tool-use RL: replace clean training trajectories with perturbation-augmented trajectories. We hypothesize that this closes the gap on the perturbation categories that are expressible as static training-data edits (observation, action, reward) but not on transition perturbations, which manifest only as runtime tool-execution responses and cannot be expressed in static data without a custom retry-aware rollout environment. We deliberately exclude transition perturbations from training: if RL on the three statically-augmentable categories alone improves Transition robustness, that is evidence of behavioral transfer rather than curriculum-specific overfitting. This setup is what allows the surprising 27% Transition gap closure we report in §6.4 to be read as transfer rather than supervision.

### 5.2. Training

We start from the publicly released ToolRL recipe (Qian et al., 2025): GRPO (Shao et al., 2024) on Qwen2.5-3B-Instruct (Qwen et al., 2025) with structured rewards on tool-name and parameter correctness. The modification is the training data: we replace the 4,000 clean rlla_4k samples with perturbation-augmented samples drawn uniformly across the 16 statically-augmentable types (4 observation + 6 action + 6 reward); transition perturbations are excluded by construction (§5.1). We deliver two checkpoints, both starting from `Qwen/Qwen2.5-3B-Instruct` with identical code and hyperparameters but different data composition. **ToolRL-DR-Full** replaces every clean trajectory with a perturbed counterpart (3,984 total: 3,905 train + 79 val). **ToolRL-DR-Mixed** keeps roughly half the

clean distribution alongside the perturbations (4,000 total: 2,006 clean / 1,994 perturbed; the same 79 validation samples are reused). A natural concern is whether replacing all clean samples degrades clean accuracy; empirically (Table 2) DR-Full's Clean accuracy ($0.643 \pm 0.065$) is statistically indistinguishable from both DR-Mixed ($0.658 \pm 0.065$) and the public ToolRL checkpoint at the same backbone ($0.638 \pm 0.065$, hereafter **ToolRL-Clean**: `chengq9/ToolRL-Qwen2.5-3B`). All other training details (GRPO objective, KL coefficient, learning rate, batch size, optimizer, hardware, wall-clock) follow the public ToolRL repository unchanged and are listed in Appendix M (Tables 16–17).

## 6. Experiments

### 6.1. Setup

We evaluate 21 models grouped into five families: (1) *RL-trained tool models*: the ToolRL family on Qwen2.5-1.5B/3B and Llama-3.2-3B (Qian et al., 2025), SFT-Clean-4k-Qwen2.5-3B, TL-CodeLLaMA-2 (Ye et al., 2024c), LoopTool 8B/32B (Zhang et al., 2025), MUA-RL 8B/14B/32B (Zhao et al., 2025); (2) *base / instruct counterparts*: Qwen2.5-1.5B/3B-Instruct, Llama-3.2-3B-Instruct, Qwen3-8B/14B/32B in non-thinking mode (Yang et al., 2025); (3) *frontier reasoning*: DeepSeek-R1-Distill-Qwen-14B (DeepSeek-AI, 2025), Qwen3.5-9B; (4) *closed-source frontier*: o4-mini via the OpenAI Chat Completions API; (5) *our trained checkpoints*: ToolRL-DR-Full and ToolRL-DR-Mixed (§5). Each model is served by vLLM (Kwon et al., 2023) at temperature 0 in the inference mode that matches its training paradigm; per-model serving and decoding parameters are in Appendix K. We evaluate the clean baseline (199 samples) and all 22 perturbation types under the generation/injection procedure of §4.3, scored by deterministic rule-based parsers (no LLM judge; Appendix I). All accuracies and drops are reported as mean $\pm$ 95% percentile-bootstrap half-width on per-sample binary correctness ($B = 10{,}000$ resamples; details in table captions).

### 6.2. Main result: an uneven sim-to-real gap

Across the 21 evaluated models, the sim-to-real gap is severe and uneven. Transition and reward perturbations cause the largest drops (per-component averages near 30% and 40% pp respectively; Table 2 columns $\Delta_{\text{Trn}}$ / $\Delta_{\text{Rew}}$), while observation perturbations are absorbed by every model ($\Delta_{\text{Obs}} < 5\%$). Action perturbations sit in the middle (10–20% pp), driven mostly by same-name distractor difficulty; a few of the largest models show negative $\Delta_{\text{Act}}$ because the distractors actually help disambiguation. Within Reward, the abbreviation variants (MisDesc-Abbr, TimeDesc-Abbr) consistently produce the largest drops; within Transition, variance across error types (Timeout vs. AuthErr vs. SchemaD) is small relative to variance across models. Per-perturbation breakdown and a heat-map view of all 21×22 (model, perturbation type) drops are in Appendices J and B.

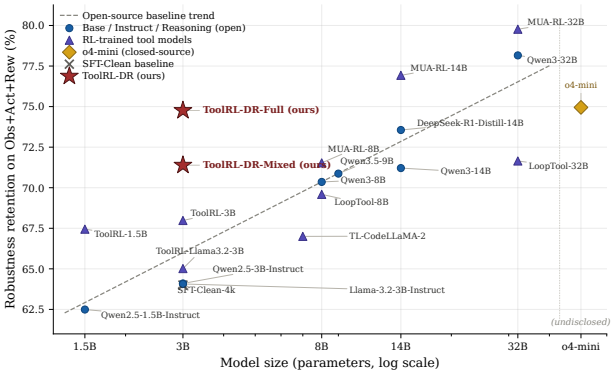



*Figure 2.* Robustness retention on Observation, Action, and Reward vs. model size, summarising Table 2. Retention $= 1 - \text{mean}(\Delta_{\text{Obs}}, \Delta_{\text{Act}}, \Delta_{\text{Rew}})/\text{Clean}$ (higher is better). Our 3B **ToolRL-DR-Full** (red star) sits above the size-matched ToolRL baseline and the log-linear trend over 18 open/closed models, comparable to the 32B function-calling models we evaluate.

**Error-mode classification.** We classify each scored-incorrect prediction by inspecting the raw model output before the format-tolerant parser (rubric in Appendix L). On transition perturbations the dominant failure modes for ToolRL-Clean (across all 808 failed transition samples in the union of the six transient-error variants) are *wrong tool call* (53.0%, 428/808; the model emitted a parseable call but with the wrong name or parameters, typically retrying the same tool with a slight argument tweak) and *omitted tool call* (47.0%, 380/808; the model produced text only, often saying "the tool seems to have failed; please try again later"). *Empty tool call* accounts for <0.1% (the model never goes silent on this benchmark); the model always produces *some* output but half the time gives up on the second-pass invocation. Our ToolRL-DR-Full reduces the omitted-call rate to 34.2% ($-12.8$ pp) while raising the wrong-call rate to 65.8% ($+12.8$ pp), with overall transition failures dropping from 808 to 702; we use this contrast to explain the transition transfer in §6.4. On reward perturbations the dominant failure is *name bias*: the model selects a distractor whose name suggests efficiency despite a description listing higher cost. Three verbatim case studies illustrating one Transition, one Reward, and one Observation failure are in Appendix O.

### 6.3. Scaling does not close the gap

We compare three Qwen3-based RL families across model sizes: MUA-RL (8B/14B/32B), LoopTool (8B/32B), Qwen3 base (8B/14B/32B), and report performance versus model size (Figure 3).

Clean accuracy improves with size within every family, as

*Table 2.* Main results across 21 models and 22 perturbation types. *Clean* (199 unperturbed samples) and *Pert. Acc.* (sample-weighted average over all perturbed samples, clean excluded) are accuracies in $[0, 1]$ (*higher* = *better*). The four $\Delta$ columns are per-POMDP-component robustness gaps on the same scale: $\Delta_X$ = Clean − mean(Acc on $X$-perturbations) (*smaller* = *better*). Each entry is mean ± half-width of the 95% percentile-bootstrap CI over per-sample correctness scores ($B = 10{,}000$; decoding is deterministic, so the only variance source is finite-sample uncertainty across benchmark items). Rows are sorted within each group by Pert. Acc. descending; **bold** marks our two trained checkpoints. Superscripts on our rows denote paired-bootstrap significance vs. the public **ToolRL-Clean** 3B baseline ($^{*}p < 0.05$, $^{**}p < 0.01$, $^{***}p < 0.001$; no marker = not significant at $p < 0.05$).

| | Clean ↑ | Pert. Acc. ↑ | $\Delta_{\text{Obs}}$ ↓ | $\Delta_{\text{Act}}$ ↓ | $\Delta_{\text{Rew}}$ ↓ | $\Delta_{\text{Trn}}$ ↓ |
|---|---|---|---|---|---|---|
| *Our 3B trained checkpoints* | | | | | | |
| **ToolRL-DR-Full (ours)** | **0.643±0.065** | **0.463±0.016**$^{***}$ | **0.009±0.075** | **0.147±0.074** | **0.331±0.076**$^{***}$ | **0.231±0.071**$^{***}$ |
| **ToolRL-DR-Mixed (ours)** | **0.658±0.065** | **0.461±0.017**$^{***}$ | **0.049±0.075** | **0.152±0.075** | **0.364±0.074**$^{***}$ | **0.232±0.072**$^{***}$ |
| *Open-source RL tool models, 8B–14B* | | | | | | |
| MUA-RL-8B | 0.678±0.065 | 0.479±0.016 | 0.023±0.073 | 0.156±0.072 | 0.401±0.073 | 0.229±0.070 |
| LoopTool-8B | 0.714±0.063 | 0.476±0.017 | 0.049±0.070 | 0.173±0.072 | 0.429±0.071 | 0.296±0.069 |
| MUA-RL-14B | 0.628±0.068 | 0.470±0.017 | 0.011±0.075 | 0.090±0.077 | 0.334±0.075 | 0.202±0.073 |
| *Open-source RL tool models, 1.5B–7B* | | | | | | |
| ToolRL-Qwen2.5-1.5B | 0.704±0.063 | 0.432±0.016 | 0.033±0.072 | 0.174±0.072 | 0.481±0.071 | 0.379±0.070 |
| TL-CodeLLaMA-2 | 0.653±0.065 | 0.405±0.016 | 0.024±0.075 | 0.166±0.073 | 0.456±0.073 | 0.333±0.072 |
| ToolRL-Qwen2.5-3B | 0.638±0.065 | 0.404±0.016 | 0.020±0.075 | 0.143±0.075 | 0.450±0.073 | 0.315±0.071 |
| ToolRL-Llama3.2-3B | 0.628±0.065 | 0.390±0.016 | 0.025±0.075 | 0.171±0.075 | 0.463±0.073 | 0.296±0.074 |
| *Open-source RL tool models, 32B* | | | | | | |
| MUA-RL-32B | 0.683±0.065 | 0.527±0.016 | 0.028±0.072 | -0.010±0.071 | 0.397±0.072 | 0.218±0.071 |
| LoopTool-32B | 0.779±0.058 | 0.508±0.017 | 0.034±0.065 | 0.120±0.066 | 0.509±0.067 | 0.396±0.063 |
| *Base / instruct counterparts* | | | | | | |
| Qwen3-14B | 0.749±0.060 | 0.529±0.016 | 0.039±0.068 | 0.143±0.069 | 0.464±0.070 | 0.250±0.067 |
| Qwen3-32B | 0.754±0.058 | 0.524±0.017 | 0.024±0.067 | 0.104±0.068 | 0.366±0.070 | 0.377±0.066 |
| Qwen3-8B | 0.673±0.065 | 0.448±0.016 | 0.019±0.075 | 0.171±0.073 | 0.409±0.072 | 0.293±0.070 |
| Qwen2.5-3B-Instruct | 0.608±0.070 | 0.390±0.016 | 0.014±0.076 | 0.195±0.075 | 0.446±0.074 | 0.237±0.073 |
| Qwen2.5-1.5B-Instruct | 0.573±0.070 | 0.367±0.016 | 0.040±0.077 | 0.200±0.077 | 0.405±0.074 | 0.205±0.073 |
| Llama-3.2-3B-Instruct | 0.518±0.070 | 0.341±0.016 | 0.009±0.077 | 0.177±0.076 | 0.372±0.073 | 0.174±0.073 |
| *Frontier (open weights, reasoning / general)* | | | | | | |
| DeepSeek-R1-Distill-14B | 0.618±0.065 | 0.442±0.016 | 0.013±0.077 | 0.160±0.075 | 0.318±0.074 | 0.213±0.073 |
| Qwen3.5-9B | 0.709±0.063 | 0.437±0.016 | 0.040±0.071 | 0.168±0.072 | 0.411±0.073 | 0.417±0.069 |
| *Closed-source frontier* | | | | | | |
| o4-mini (OpenAI) | 0.709±0.063 | 0.502±0.017 | 0.049±0.071 | 0.157±0.071 | 0.326±0.072 | 0.276±0.070 |
| *ToolRL SFT version* | | | | | | |
| SFT-Clean-4k-3B | 0.598±0.068 | 0.267±0.014 | 0.019±0.075 | 0.184±0.076 | 0.443±0.074 | 0.577±0.068 |

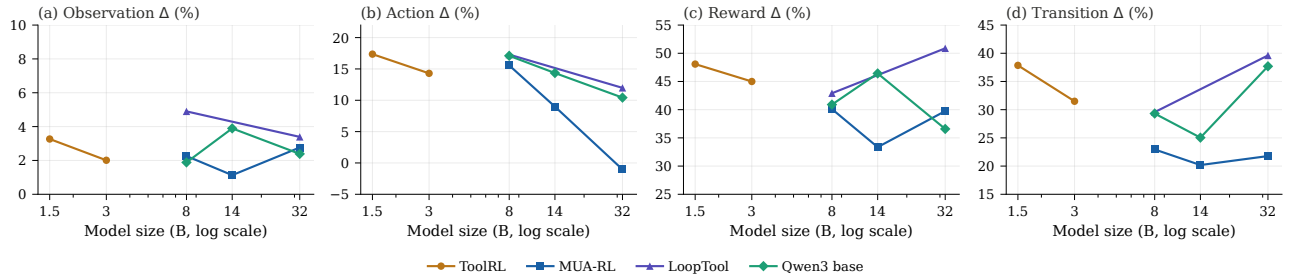



*Figure 3.* Per-POMDP-component drop $\Delta$ vs. model size for four families (ToolRL 1.5B/3B, MUA-RL 8B/14B/32B, LoopTool 8B/32B, Qwen3 base 8B/14B/32B). Smaller $\Delta$ = better. Observation drops stay small (<5%) and Action drops trend slightly downward with size, but Reward (c) and Transition (d) stay within the same band as size grows from 8B to 32B, motivating training-side interventions in §5.

expected. The drops, however, are largely flat or even *increase* with size for transition and reward perturbations. LoopTool-32B has a 50.9% Reward drop and 39.6% Transition drop, larger than its 8B sibling on both axes; Qwen3-32B has a 37.7% Transition drop, larger than Qwen3-8B's 29.3%. The MUA-RL family is the only one in which the largest model has the smallest drops, but even there the residual reward and transition gaps are 40% and 22%. Robustness, in this taxonomy, is not a small-model artifact: 32B-scale RL-trained tool models still fail in the same patterns as their 8B siblings. SFT-Clean-4k-Qwen2.5-3B (Table 2, the supervised counterpart of ToolRL at the same backbone)

has a Transition drop ($\Delta_{\mathrm{Trn}} = 57.7\%$) roughly $2\times$ that of any RL-trained model, a gap that motivates rollout-based training in general.

### 6.4. Domain randomization narrows the gap

**Training-distribution comparison on the 3B backbone.** **ToolRL-Clean**, our **ToolRL-DR-Full**, and our **ToolRL-DR-Mixed** share the Qwen2.5-3B-Instruct backbone and differ only in the GRPO training distribution (§5). ToolRL-DR-Full closes Reward by 11.9 pp (≈26% of ToolRL-Clean's Reward gap) and *unexpectedly* closes Transition by 8.4 pp (≈27%), despite no Transition perturbation in training. This is our most surprising finding: perturbed-RL training transfers some robustness to runtime errors it never saw. DR-Mixed lies between Clean and DR-Full on Reward and Obs, reproducing a dose response: more perturbation in training $\rightarrow$ smaller drops at evaluation. The Action axis is invariant across all three RL conditions, suggesting same-name distractor disambiguation is bounded by model capacity rather than training-data composition. Figure 2 visualizes this against the broader 1.5B–32B trend on the Obs+Act+Rew axes.

**Why Transition transfers: a more persistent retry policy.** We attribute transfer to a more persistent retry policy under adversarial inputs. Re-running the failure-mode classification of §6.2 on ToolRL-DR-Full's six transient-error variants yields 702 failed transition samples (down from 808 for ToolRL-Clean), with omitted-call rate dropping from 47.0% to 34.2% ($-12.8$ pp) and wrong-call rate rising from 53.0% to 65.8%. DR-Full converts some giveups into successful retries (total failures down 13%) and retries more often when it does fail. We do not claim the gap is closed: $\Delta_{\mathrm{Trn}} \approx 0.23$ remains, and a retry-aware reward (e.g. rollouts with error responses (Snell et al., 2024; Brown et al., 2024)) is a natural next step.

### 6.5. Live evaluation platform

A public RobustBench-TC leaderboard hosted as a HuggingFace Space (URL withheld for double-blind review; Appendix P) lets contributors upload predictions JSONs that are scored server-side with the same deterministic scorer used in this paper (§6.1); and it is seeded with the 21 evaluated models.

## 7. Limitations and future work

**Multi-turn coverage.** Our evaluations focus on single-turn settings. Multi-turn agents can sometimes recover via follow-up clarification but also accumulate errors over horizon, and transition perturbations are hard to inject cleanly into multi-turn rollouts that already contain tool-execution loops. The perturbation taxonomy carries over directly to future extension to multi-turn.

**Method scale.** In this work, ToolRL-DR is trained on a 3B backbone with 4,000 samples to match the public ToolRL recipe and enable apples-to-apples comparison. Whether the recipe scales to 14B/32B without saturation, and whether more training data improves headroom, are open.

**Transition robustness.** Our method does not close the Transition gap by design. Doing so likely requires either RL with rollouts that include error responses and a retry-aware reward, or test-time strategies (self-consistency, retry policies) that operate independently of training.

## 8. Conclusion

We frame tool-use robustness as a sim-to-real problem and introduce a benchmark that perturbs the observation, action, reward-relevant metadata, and transition components of the tool-use POMDP. Across 21 models, the resulting gap is large and uneven: observation perturbations are mostly absorbed, whereas reward-relevant and transition perturbations remain difficult, and increasing model size does not consistently close these gaps in our evaluation. We further show that domain-randomized RL on perturbation-augmented tool-use trajectories improves robustness on a 3B backbone, with the performance improvement on reward-relevant perturbations and partial transfer to held-out transition failures. Notably, our 3B checkpoint closes approximately 27% of the Transition gap without ever training on transition perturbations, suggesting that RL on adversarial static inputs induces a more persistent retry policy that generalizes to runtime failures. These results suggest that tool-use reliability should be evaluated not only under clean tool registries and deterministic execution, but also under deployment-style metadata noise, action-space ambiguity, and runtime errors.

## References

Akkaya, I., Andrychowicz, M., Chociej, M., Litwin, M., McGrew, B., Petron, A., Paino, A., Plappert, M., Powell, G., Ribas, R., et al. Solving rubik's cube with a robot hand, 2019.

Brown, B., Juravsky, J., Ehrlich, R., Clark, R., Le, Q. V., Ré, C., and Mirhoseini, A. Large language monkeys: Scaling inference compute with repeated sampling, 2024.

Chebotar, Y., Handa, A., Makoviychuk, V., Macklin, M., Issac, J., Ratliff, N., and Fox, D. Closing the sim-to-real loop: Adapting simulation randomization with real world experience. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2019.

Chen, C., Hao, X., Liu, W., Huang, X., Zeng, X., Yu, S., Li,

D., Huang, Y., Liu, X., Xinzhi, W., et al. Acebench: A comprehensive evaluation of llm tool usage. *Findings of the Association for Computational Linguistics: EMNLP*, 2025:12970–12998, 2025.

crystaldba contributors. crystaldba/postgres-mcp PR #157: Disambiguation clauses for sibling tools, 2025. URL https://github.com/crystaldba/postgres-mcp/pull/157.

DeepSeek-AI. DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning, 2025.

Faghih, K., Wang, W., Cheng, Y., Bharti, S., Sriramanan, G., Balasubramanian, S., Hosseini, P., and Feizi, S. Tool preferences in agentic llms are unreliable. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 20965–20980, 2025.

Gorilla contributors. Gorilla/BFCL issue #839: vllm server disconnects mid-inference, 2025. URL https://github.com/ShishirPatil/gorilla/issues/839.

Grafana Loki MCP contributors. grafana/loki-mcp issue #27: Parameter description "1h ago" fails parser, 2025. URL https://github.com/grafana/loki-mcp/issues/27.

Höfer, S., Bekris, K., Handa, A., Gamboa, J. C., Mozifian, M., Golemo, F., Atkeson, C., Fox, D., Goldberg, K., Leonard, J., et al. Sim2Real in robotics and automation: Applications and challenges. *IEEE Transactions on Automation Science and Engineering*, 2021.

Kwon, W., Li, Z., Zhuang, S., Sheng, Y., Zheng, L., Yu, C. H., Gonzalez, J. E., Zhang, H., and Stoica, I. Efficient memory management for large language model serving with PagedAttention. In *Proceedings of the 29th Symposium on Operating Systems Principles (SOSP)*, 2023.

LangChain contributors. langchain issue #29596: Missing authorization header causes silent 401, 2025a. URL https://github.com/langchain-ai/langchain/issues/29596.

LangChain contributors. langchain issue #34746: Ollama returns malformed json; tool call dropped, 2025b. URL https://github.com/langchain-ai/langchain/issues/34746.

LangChain contributors. langchain issue #35597: Default `request_timeout=None` causes agent hang, 2025c. URL https://github.com/langchain-ai/langchain/issues/35597.

LangChain contributors. langchain issue #36032: `anyOf` schema crashes ollama after definition update, 2025d. URL https://github.com/langchain-ai/langchain/issues/36032.

Li, M., Zhao, Y., Yu, B., Song, F., Li, H., Yu, H., Li, Z., Huang, F., and Li, Y. API-Bank: A comprehensive benchmark for tool-augmented LLMs. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2023.

Liu, Z., Hoang, T., Zhang, J., Zhu, M., Lan, T., Kokane, S., Tan, J., Yao, W., Liu, Z., Feng, Y., et al. APIGen: Automated pipeline for generating verifiable and diverse function-calling datasets, 2024.

LlamaIndex contributors. LlamaIndex issue #7170: Tool name typo from user query crashes dispatcher, 2023. URL https://github.com/run-llama/llama_index/issues/7170.

LlamaIndex contributors. LlamaIndex issue #16757: Query paraphrase routes to wrong tool, 2024. URL https://github.com/run-llama/llama_index/issues/16757.

Lu, J., Holleis, T., Zhang, Y., Aumayer, B., Nan, F., Bai, H., Ma, S., Ma, S., Li, M., Yin, G., et al. Toolsandbox: A stateful, conversational, interactive evaluation benchmark for llm tool use capabilities. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pp. 1160–1183, 2025.

Microsoft Semantic Kernel contributors. microsoft/semantic-kernel issue #13690: Silent mid-session tool swap with abbreviated descriptions, 2025. URL https://github.com/microsoft/semantic-kernel/issues/13690.

NetBox Labs. netbox-mcp-server issue #79: Misleading filter description silently returns all records, 2025. URL https://github.com/netboxlabs/netbox-mcp-server/issues/79.

OpenAI. GPT-4o-mini model specification, 2024. URL https://platform.openai.com/docs/models/gpt-4o-mini.

OpenAI. GPT-5-mini model specification, 2025. URL https://platform.openai.com/docs/models/gpt-5-mini.

openai-agents-python contributors. openai-agents-python issue #1167: Same-named tools across mcp servers cause sdk hang, 2025. URL https://github.com/openai/openai-agents-python/issues/1167.

OpenAI Python contributors. openai-python issue #2699: Rate-limit asymmetry across endpoints, 2025. URL https://github.com/openai/openai-python/issues/2699.

Patil, S. G., Zhang, T., Wang, X., and Gonzalez, J. E. Gorilla: Large language model connected with massive apis. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Peng, X. B., Andrychowicz, M., Zaremba, W., and Abbeel, P. Sim-to-real transfer of robotic control with dynamics randomization. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2018.

Qian, C., Acikgoz, E. C., He, Q., Wang, H., Chen, X., Hakkani-Tür, D., Tur, G., and Ji, H. ToolRL: Reward is all tool learning needs, 2025.

Qin, Y., Liang, S., Ye, Y., Zhu, K., Yan, L., Lu, Y., Lin, Y., Cong, X., Tang, X., Qian, B., et al. ToolLLM: Facilitating large language models to master 16000+ real-world APIs. In *Proceedings of the 12th International Conference on Learning Representations (ICLR)*, 2024.

Qwen, :, Yang, A., Yang, B., Zhang, B., Hui, B., Zheng, B., Yu, B., Li, C., Liu, D., Huang, F., Wei, H., Lin, H., Yang, J., Tu, J., Zhang, J., Yang, J., Yang, J., Zhou, J., Lin, J., Dang, K., Lu, K., Bao, K., Yang, K., Yu, L., Li, M., Xue, M., Zhang, P., Zhu, Q., Men, R., Lin, R., Li, T., Tang, T., Xia, T., Ren, X., Ren, X., Fan, Y., Su, Y., Zhang, Y., Wan, Y., Liu, Y., Cui, Z., Zhang, Z., and Qiu, Z. Qwen2.5 technical report, 2025. URL https://arxiv.org/abs/2412.15115.

Rabinovich, E. and Anaby-Tavor, A. On the robustness of agentic function calling, 2025.

Raj, H., Rosati, D., and Majumdar, S. Measuring reliability of large language models through semantic consistency, 2022.

Sadeghi, F. and Levine, S. CAD$^2$RL: Real single-image flight without a single real image. In *Proceedings of Robotics: Science and Systems (RSS)*, 2017.

Schick, T., Dwivedi-Yu, J., Dessì, R., Raileanu, R., Lomeli, M., Zettlemoyer, L., Cancedda, N., and Scialom, T. Toolformer: Language models can teach themselves to use tools. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Shao, Z., Wang, P., Zhu, Q., Xu, R., Song, J., Bi, X., Zhang, H., Zhang, M., Li, Y. K., Wu, Y., et al. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models, 2024.

Sierra Research. tau-bench issue #39: Tool description vs. implementation mismatch, 2025. URL https://github.com/sierra-research/tau-bench/issues/39.

Snell, C., Lee, J., Xu, K., and Kumar, A. Scaling LLM test-time compute optimally can be more effective than scaling model parameters, 2024.

Tan, J., Zhang, T., Coumans, E., Iscen, A., Bai, Y., Hafner, D., Bohez, S., and Vanhoucke, V. Sim-to-real: Learning agile locomotion for quadruped robots. In *Proceedings of Robotics: Science and Systems (RSS)*, 2018.

Tang, Q., Deng, Z., Lin, H., Han, X., Liang, Q., Cao, B., and Sun, L. ToolAlpaca: Generalized tool learning for language models with 3000 simulated cases, 2023.

Tobin, J., Fong, R., Ray, A., Schneider, J., Zaremba, W., and Abbeel, P. Domain randomization for transferring deep neural networks from simulation to the real world. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017.

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

Yang, A., Li, A., Yang, B., Zhang, B., Hui, B., Zheng, B., Yu, B., Gao, C., Huang, C., Lv, C., et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K., and Cao, Y. ReAct: Synergizing reasoning and acting in language models. In *Proceedings of the 11th International Conference on Learning Representations (ICLR)*, 2023.

Yao, S., Shinn, N., Razavi, P., and Narasimhan, K. $\tau$-bench: A benchmark for tool-agent-user interaction in real-world domains, 2024.

Ye, J., Li, G., Gao, S., Huang, C., Wu, Y., et al. ToolEyes: Fine-grained evaluation for tool learning capabilities of large language models in real-world scenarios, 2024a.

Ye, J., Wu, Y., Gao, S., Huang, C., Li, S., Li, G., Fan, X., Zhang, Q., Gui, T., and Huang, X.-J. Rotbench: A multi-level benchmark for evaluating the robustness of large language models in tool learning. In *Proceedings of the 2024 conference on empirical methods in natural language processing*, pp. 313–333, 2024b.

Ye, J., Wu, Y., Li, S., Yang, Y., Xi, Z., Gui, T., Zhang, Q., Huang, X., Wang, P., Shi, Z., et al. Tl-training: A task-feature-based framework for training large language

models in tool use. *arXiv preprint arXiv:2412.15495*, 2024c.

Zhang, K., Jiao, W., Du, K., Lu, Y., Liu, W., Zhang, W., and Yu, Y. Looptool: Closing the data-training loop for robust llm tool calls. *arXiv preprint arXiv:2511.09148*, 2025.

Zhao, W., Queralta, J. P., and Westerlund, T. Sim-to-real transfer in deep reinforcement learning for robotics: A survey. In *IEEE Symposium Series on Computational Intelligence (SSCI)*, 2020.

Zhao, W., Wang, X., Ma, C., Kong, L., Yang, Z., Tuo, M., Shi, X., Zhai, Y., and Cai, X. Mua-rl: Multi-turn user-interacting agent reinforcement learning for agentic tool use. *arXiv preprint arXiv:2508.18669*, 2025.

## A. Per-source benchmark statistics

Table 3 gives the full per-source composition of RobustBench-TC: clean sub-sample counts, average evaluable samples per perturbation type, totals, average candidate-tool counts, and the structured output format expected by each source's original scorer.

*Table 3.* Per-source composition of RobustBench-TC. *Clean* is the sub-sample count we use as baseline. *Avg./type* is the mean evaluable-sample count per (source, perturbation type) pair, smaller than *Clean* when some perturbations do not apply (only ToolEyes hits this, on the 12 Action and Reward types). *Total* is the per-source sum across the clean baseline plus all 22 perturbation types, aggregating to 3,721 samples. *Avg. tools* is the mean candidate-tool count per sample (clean slice). *Output format* lists the structured form expected by the original scorer.

| Benchmark | Clean | Avg./type | Total | Avg. tools | Output format |
|---|---|---|---|---|---|
| BFCL V3 (ST) | 32 | 22.0 | 505 | 2.8 | *[func(a=1)]* (BFCL AST) |
| API-Bank | 74 | 71.6 | 1646 | 3.1 | *¡tool_call¿{json}¡/tool_call¿* |
| RoTBench | 21 | 20.8 | 479 | 7.8 | *Action: X / Action Input: {json}* |
| ToolAlpaca | 21 | 20.8 | 479 | 3.5 | mixed (ReAct, XML, JSON, bare Python) |
| ToolEyes | 51 | 25.5 | 612 | 6.3 | ReAct |
| **Total** | **199** | **160.0** | **3721** | — | — |

## B. Per-(model, perturbation) drop heatmap

Figure 4 consolidates the per-perturbation breakdown (Appendix J) into one heatmap, showing every (model, perturbation) accuracy drop on a single page. Rows are the 21 evaluated models grouped by family; columns are the 22 perturbation types grouped by POMDP component (white vertical lines). Cell color and annotation are the percentage-point drop in accuracy from the model's own clean baseline (larger = more brittle).

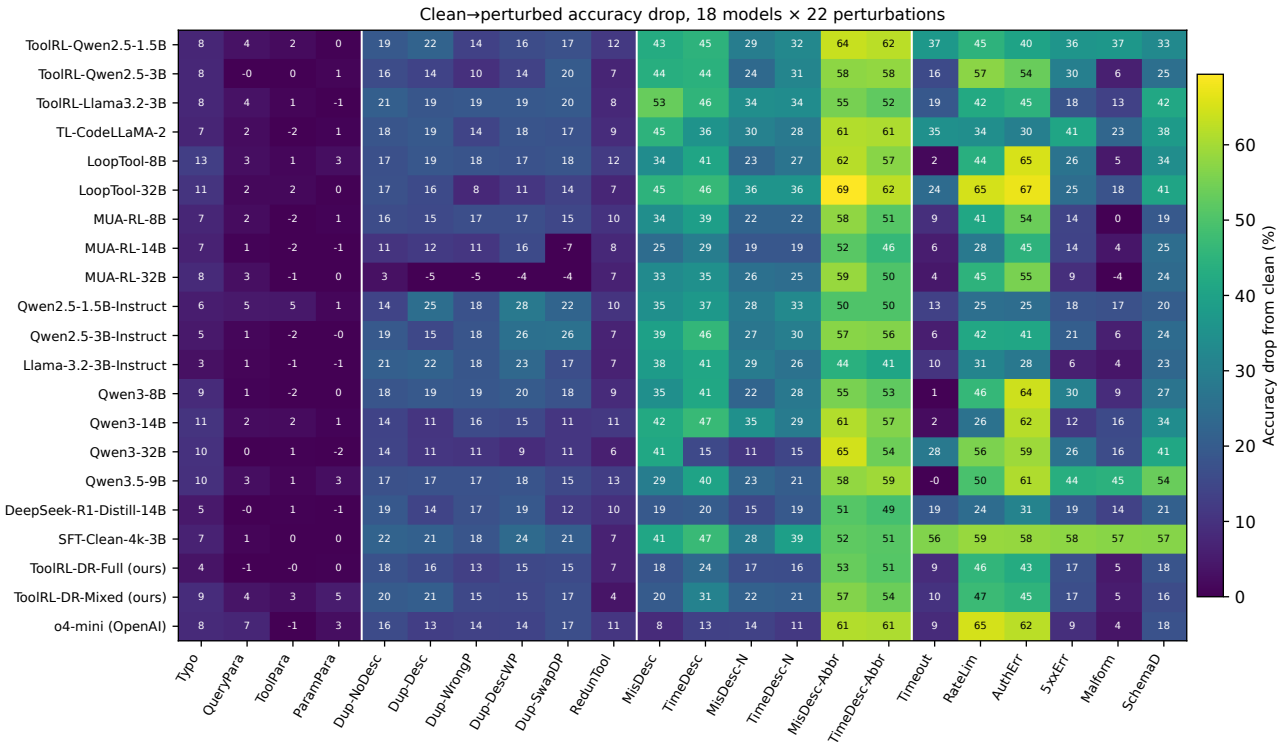



*Figure 4.* Per-(model, perturbation) accuracy drop from clean across all 21 evaluated models and 22 perturbations. Rows: model families. Columns: 22 perturbation types grouped by POMDP component (white vertical lines). Cell color and annotation = percentage-point drop in accuracy from the model's own clean baseline. The visual takeaway is that drops are concentrated on the right two POMDP components (Reward, Transition) regardless of model size or family, and that our ToolRL-DR-Full / -Mixed rows show smaller drops on the Reward block than every other 3B model.

## C. One example per perturbation

Each perturbation type is illustrated below by a clean source sample (drawn from the corresponding source benchmark) paired with its perturbed counterpart, so that the reader can inspect what each type actually changes in the data.

Each subtype below is illustrated with one canonical (clean, perturbed) pair drawn from the released dataset. We pick the same sample id across perturbations where possible to make the visual diff easier; transition perturbations operate on clean samples at runtime and are illustrated with the injected error string.

### Typo *(Observation, code: `realistic_typos`)*

**Sample id:** `bfcl_v3__BFCL_v3_multiple__multiple_110` (*bfcl_v3*)

- **User query (clean, divergence):** *Find the type of gene mutation based on SNP (Single Nucleotide Polymorphism) ID rs6034464.*
- **User query (perturbed, divergence):** *Find teh type of gene mutaiton based on SNP (Single Nucleotide Polymorpism) ID rs6034464.*
- **Tools (unchanged):** `get_collectables_in_season, mutation_type.find`
- **GT (clean):** `[{"name": "mutation_type.find", "parameters": {"snp_id": "rs6034464", "species": "Homo sapiens"}}]`

### QueryPara *(Observation, code: `query_paraphrase`)*

**Sample id:** `bfcl_v3__BFCL_v3_multiple__multiple_110` (*bfcl_v3*)

- **User query (clean, divergence):** *Find the type of gene mutation based on SNP (Single Nucleotide Polymorphism) ID*

*rs6034464.*

- **User query (perturbed, divergence):** *Identify the kind of gene mutation associated with SNP (Single Nucleotide Polymorphism) ID rs6034464.*
- **Tools (unchanged):** `get_collectables_in_season, mutation_type.find`
- **GT (clean):** `[{"name": "mutation_type.find", "parameters": {"snp_id": "rs6034464", "species": "Homo sapiens"}}]`

**ToolPara** ***(Observation, code: `paraphrase_tool_description`)***

**Sample id:** `bfcl_v3__BFCL_v3_multiple__multiple_110` (*bfcl_v3*)

- **User query:** *Find the type of gene mutation based on SNP (Single Nucleotide Polymorphism) ID rs6034464.*
- **Tools (unchanged):** `get_collectables_in_season, mutation_type.find`
- **Tool entry (clean):** `{"name": "get_collectables_in_season", "description": "Retrieve a list of collectable items in a specific game during a specified season.", "parameters": {"type": "dict", "properties": {"game_name": {"type": "string", "description": "Name of the game."}, "season": {"type": "strin`
- **Tool entry (perturbed):** `{"name": "get_collectables_in_season", "description": "Obtain a list of collectible items available in a certain game for a designated season.", "parameters": {"type": "dict", "properties": {"game_name": {"type": "string", "description": "Name of the game."}, "season": {"type": "`
- **GT (clean):** `[{"name": "mutation_type.find", "parameters": {"snp_id": "rs6034464", "species": "Homo sapiens"}}]`

**ParamPara** ***(Observation, code: `paraphrase_parameter_description`)***

**Sample id:** `bfcl_v3__BFCL_v3_multiple__multiple_110` (*bfcl_v3*)

- **User query:** *Find the type of gene mutation based on SNP (Single Nucleotide Polymorphism) ID rs6034464.*
- **Tools (unchanged):** `get_collectables_in_season, mutation_type.find`
- **Tool entry (clean):** `{"name": "get_collectables_in_season", "description": "Retrieve a list of collectable items in a specific game during a specified season.", "parameters": {"type": "dict", "properties": {"game_name": {"type": "string", "description": "Name of the game."}, "season": {"type": "strin`
- **Tool entry (perturbed):** `{"name": "get_collectables_in_season", "description": "Retrieve a list of collectable items in a specific game during a specified season.", "parameters": {"type": "dict", "properties": {"game_name": {"type": "string", "description": "Title of the game."}, "season": {"type": "stri`
- **GT (clean):** `[{"name": "mutation_type.find", "parameters": {"snp_id": "rs6034464", "species": "Homo sapiens"}}]`

**Dup-NoDesc** ***(Action, code: `same_name_A`)***

**Sample id:** `apibank__level1_101` (*apibank*)

- **User query:** ***Dialogue Records History** ¡user¿Can you help me modify an alarm for user3 at 2023-03-24 09:00:00?¡/user¿ ¡response¿Sure, to modify an alarm, I need to authenticate the user. Can...*
- **Tools (clean):** `ModifyAlarm, GetUserToken, AddAgenda, **Think`
- **Tools (perturbed):** `ModifyAlarm, GetUserToken, AddAgenda, AddAgenda, **Think`
- **GT (clean):** `[{"name": "AddAgenda", "parameters": {"token": "p9o8i7u6y5t4r3e2w1q", "content": "Lunch with friends", "time": "2023-03-24 14:00:00", "location": "Restaurant X"}}]`

**Dup-Desc** ***(Action, code: `same_name_B`)***

**Sample id:** `apibank__level1_101` (*apibank*)

- **User query:** ***Dialogue Records History** ¡user¿Can you help me modify an alarm for user3 at 2023-03-24 09:00:00?¡/user¿ ¡response¿Sure, to modify an alarm, I need to authenticate the user. Can...*
- **Tools (clean):** `ModifyAlarm, GetUserToken, AddAgenda, **Think`
- **Tools (perturbed):** `ModifyAlarm, GetUserToken, AddAgenda, AddAgenda, **Think`
- **GT (clean):** `[{"name": "AddAgenda", "parameters": {"token": "p9o8i7u6y5t4r3e2w1q", "content": "Lunch with friends", "time": "2023-03-24 14:00:00", "location": "Restaurant X"}}]`

**Dup-WrongP** ***(Action, code: `same_name_C`)***

**Sample id:** `apibank__level1_101` (*apibank*)

- **User query:** ***Dialogue Records History** ¡user¿Can you help me modify an alarm for user3 at 2023-03-24 09:00:00?¡/user¿ ¡response¿Sure, to modify an alarm, I need to authenticate the user. Can...*
- **Tools (clean):** `ModifyAlarm, GetUserToken, AddAgenda, **Think`
- **Tools (perturbed):** `ModifyAlarm, AddAgenda, GetUserToken, AddAgenda, **Think`
- **GT (clean):** `[{"name": "AddAgenda", "parameters": {"token": "p9o8i7u6y5t4r3e2w1q", "content": "Lunch with friends", "time": "2023-03-24 14:00:00", "location": "Restaurant X"}}]`

**Dup-DescWP** ***(Action, code: `same_name_D`)***

**Sample id:** `apibank__level1_101` (*apibank*)

- **User query:** ***Dialogue Records History** ¡user¿Can you help me modify an alarm for user3 at 2023-03-24 09:00:00?¡/user¿ ¡response¿Sure, to modify an alarm, I need to authenticate the user. Can...*
- **Tools (clean):** `ModifyAlarm, GetUserToken, AddAgenda, **Think`
- **Tools (perturbed):** `ModifyAlarm, GetUserToken, AddAgenda, AddAgenda, **Think`
- **GT (clean):** `[{"name": "AddAgenda", "parameters": {"token": "p9o8i7u6y5t4r3e2w1q", "content": "Lunch with friends", "time": "2023-03-24 14:00:00", "location": "Restaurant X"}}]`

**Dup-SwapDP** ***(Action, code: `same_name_E`)***

**Sample id:** `apibank__level1_101` (*apibank*)

- **User query:** ***Dialogue Records History** ¡user¿Can you help me modify an alarm for user3 at 2023-03-24 09:00:00?¡/user¿ ¡response¿Sure, to modify an alarm, I need to authenticate the user. Can...*
- **Tools (clean):** `ModifyAlarm, GetUserToken, AddAgenda, **Think`
- **Tools (perturbed):** `ModifyAlarm, GetUserToken, AddAgenda, AddAgenda, **Think`
- **GT (clean):** `[{"name": "AddAgenda", "parameters": {"token": "p9o8i7u6y5t4r3e2w1q", "content": "Lunch with friends", "time": "2023-03-24 14:00:00", "location": "Restaurant X"}}]`

**RedunTool** ***(Action, code: `redundant`)***

**Sample id:** `bfcl_v3__BFCL_v3_multiple__multiple_110` (*bfcl_v3*)

- **User query:** *Find the type of gene mutation based on SNP (Single Nucleotide Polymorphism) ID rs6034464.*
- **Tools (clean):** `get_collectables_in_season, mutation_type.find`
- **Tools (perturbed):** `get_collectables_in_season, mutation_type.find, get_collectables_for_player, get_collectables_in_region, mutation_type.annotate, mutation_type.predict_effect`
- **GT (clean):** `[{"name": "mutation_type.find", "parameters": {"snp_id": "rs6034464", "species": "Homo sapiens"}}]`

**MisDesc** ***(Reward, code: CD)***

**Sample id:** `apibank__level1_101` (*apibank*)

- **User query (clean, divergence):** *onse¿ ¡user¿The content is "Lunch with friends" and location is "Restaurant X".¡/user¿ ¡response¿Great! Adding agenda...¡/response¿ ¡user¿ Based on our convers*
- **User query (perturbed, divergence):** *onse¿ ¡user¿The content is "Lunch with friends" and location is "Restaurant X". Please use a cost-effective option.¡/user¿ ¡response¿Great! Adding agenda...¡/response¿ ¡user¿ Based on our convers*
- **Tools (clean):** `ModifyAlarm, GetUserToken, AddAgenda, **Think`
- **Tools (perturbed):** `AddAgenda_1, **Think, AddAgenda, GetUserToken, ModifyAlarm`
- **GT (clean):** `[{"name": "AddAgenda", "parameters": {"token": "p9o8i7u6y5t4r3e2w1q", "content": "Lunch with friends", "time": "2023-03-24 14:00:00", "location": "Restaurant X"}}]`

**TimeDesc** ***(Reward, code: TD)***

**Sample id:** `apibank__level1_101` (*apibank*)

- **User query (clean, divergence):** *onse¿ ¡user¿The content is "Lunch with friends" and location is "Restaurant X".¡/user¿ ¡response¿Great! Adding agenda...¡/response¿ ¡user¿ Based on our convers*
- **User query (perturbed, divergence):** *onse¿ ¡user¿The content is "Lunch with friends" and location is "Restaurant X". Please use the quickest method.¡/user¿ ¡response¿Great! Adding agenda...¡/response¿ ¡user¿ Based on our convers*
- **Tools (clean):** `ModifyAlarm, GetUserToken, AddAgenda, **Think`
- **Tools (perturbed):** `AddAgenda_1, ModifyAlarm, AddAgenda, **Think, GetUserToken`
- **GT (clean):** `[{"name": "AddAgenda", "parameters": {"token": "p9o8i7u6y5t4r3e2w1q", "content": "Lunch with friends", "time": "2023-03-24 14:00:00", "location": "Restaurant X"}}]`

**MisDesc-N** ***(Reward, code: CD_NT)***

**Sample id:** `apibank__level1_101` (*apibank*)

- **User query (clean, divergence):** *onse¿ ¡user¿The content is "Lunch with friends" and location is "Restaurant X".¡/user¿ ¡response¿Great! Adding agenda...¡/response¿ ¡user¿ Based on our convers*
- **User query (perturbed, divergence):** *onse¿ ¡user¿The content is "Lunch with friends" and location is "Restaurant X". Use the cheapest available tool if possible.¡/user¿ ¡response¿Great! Adding agenda...¡/response¿ ¡user¿ Based on our convers*
- **Tools (clean):** `ModifyAlarm, GetUserToken, AddAgenda, **Think`
- **Tools (perturbed):** `ModifyAlarm, GetUserToken, **Think, AddAgenda_1, AddAgenda`
- **GT (clean):** `[{"name": "AddAgenda", "parameters": {"token": "p9o8i7u6y5t4r3e2w1q", "content": "Lunch with friends", "time": "2023-03-24 14:00:00", "location": "Restaurant X"}}]`

**TimeDesc-N** ***(Reward, code: TD_NT)***

**Sample id:** `apibank__level1_101` (*apibank*)

- **User query (clean, divergence):** *onse¿ ¡user¿The content is "Lunch with friends" and location is "Restaurant X".¡/user¿ ¡response¿Great! Adding agenda...¡/response¿ ¡user¿ Based on our convers*
- **User query (perturbed, divergence):** *onse¿ ¡user¿The content is "Lunch with friends" and location is "Restaurant X". Please use the quickest method.¡/user¿ ¡response¿Great! Adding agenda...¡/response¿ ¡user¿ Based on our convers*
- **Tools (clean):** `ModifyAlarm, GetUserToken, AddAgenda, **Think`
- **Tools (perturbed):** `ModifyAlarm, **Think, AddAgenda_1, GetUserToken, AddAgenda`
- **GT (clean):** `[{"name": "AddAgenda", "parameters": {"token": "p9o8i7u6y5t4r3e2w1q", "content": "Lunch with friends", "time": "2023-03-24 14:00:00", "location": "Restaurant X"}}]`

**MisDesc-Abbr** ***(Reward, code: CD_AB)***

**Sample id:** `bfcl_v3__BFCL_v3_multiple__multiple_110` (*bfcl_v3*)

- **User query:** *Find the type of gene mutation based on SNP (Single Nucleotide Polymorphism) ID rs6034464.*
- **Tools (clean):** `get_collectables_in_season, mutation_type.find`
- **Tools (perturbed):** `mutation_type.find, get_collectables_in_season, mut_type.find`
- **GT (clean):** `[{"name": "mutation_type.find", "parameters": {"snp_id": "rs6034464", "species": "Homo sapiens"}}]`
- **GT (perturbed):** `[{"name": "mut_type.find", "parameters": {"snp_id": "rs6034464", "species": "Homo sapiens"}}]`

**TimeDesc-Abbr** ***(Reward, code: TD_AB)***

**Sample id:** `bfcl_v3__BFCL_v3_multiple__multiple_110` (*bfcl_v3*)

- **User query:** *Find the type of gene mutation based on SNP (Single Nucleotide Polymorphism) ID rs6034464.*
- **Tools (clean):** `get_collectables_in_season, mutation_type.find`
- **Tools (perturbed):** `mutation_type.find, get_collectables_in_season, mut_type.find`
- **GT (clean):** `[{"name": "mutation_type.find", "parameters": {"snp_id": "rs6034464", "species": "Homo sapiens"}}]`
- **GT (perturbed):** `[{"name": "mut_type.find", "parameters": {"snp_id": "rs6034464", "species": "Homo sapiens"}}]`

**Transition perturbations**

All six transition variants operate on the same clean dataset (the GT and tool list are unchanged); the perturbation is a fixed error string injected as the simulated tool response between pass 1 and pass 2 of inference. We illustrate with a single clean sample below; the error string is the only thing that varies per subtype. **Representative clean sample:** `bfcl_v3__BFCL_v3_multiple__multiple_2` (bfcl_v3). User query: *What is the capital of Brazil?*; tools: `country_info.largest_city, country_info.capital, country_info.population`; GT: `[{"name": "country_info.capital", "parameters": {"country": "Brazil"}}]`.

- **Timeout (transient_timeout)**: *"Tool execution timed out after the configured request timeout. The remote endpoint did not respond within the allotted time."*
- **RateLim (transient_rate_limit)**: *"HTTP 429 Too Many Requests. The provider rejected the call because the per-minute rate limit has been exceeded."*
- **AuthErr (transient_auth_error)**: *"HTTP 401 Unauthorized. The provider rejected the call because the supplied credentials are invalid or expired."*
- **5xxErr (transient_server_error)**: *"HTTP 500 Internal Server Error. The remote endpoint failed to handle the request."*
- **Malform (transient_malformed_response)**: *"Malformed response from tool execution: the body could not be parsed as JSON."*
- **SchemaD (transient_schema_drift)**: *"Schema validation failed: the response did not match the tool's declared output schema (extra/missing fields)."*

## D. Display-name to code-name mapping

Table 4 maps the abbreviated display names used in figures and tables (§4.1) to the full identifiers used in our released JSONL files.

*Table 4.* Display name ↔ code-level identifier mapping. The latter is the perturbation type field that appears in each released *api_eval/¡perturbation¿.jsonl* file as well as in every **.predictions.jsonl* record's *perturbation.type* key.

| Display name | Code-level identifier (released dataset) |
|---|---|
| Typo | realistic_typos |
| QueryPara | query_paraphrase |
| ToolPara | paraphrase_tool_description |
| ParamPara | paraphrase_parameter_description |
| Dup-NoDesc | same_name_A |
| Dup-Desc | same_name_B |
| Dup-WrongP | same_name_C |
| Dup-DescWP | same_name_D |
| Dup-SwapDP | same_name_E |
| RedunTool | redundant |
| MisDesc | CD |
| TimeDesc | TD |
| MisDesc-N | CD_NT |
| TimeDesc-N | TD_NT |
| MisDesc-Abbr | CD_AB |
| TimeDesc-Abbr | TD_AB |
| Timeout | transient_timeout |
| RateLim | transient_rate_limit |
| AuthErr | transient_auth_error |
| 5xxErr | transient_server_error |
| Malform | transient_malformed_response |
| SchemaD | transient_schema_drift |

## E. Verified production-failure evidence

§4.2 of the main paper summarises a few representative grounding examples in prose. Table 5 below is the full type → canonical-issue map for all 22 perturbation types; the rest of this appendix gives the *audit history* (which entries we initially considered, which we kept, which we demoted after re-verification, and what replaced them).

*Table 5.* Production grounding for each perturbation type. We give one canonical verified GitHub issue per type. All cited repositories are tool-calling agent frameworks, MCP servers, or tool-use benchmarks (no vision-model or general-LLM bugs).

| Type | Source | Failure summary |
|---|---|---|
| **Observation perturbations** (4 types) | | |
| Typo | LlamaIndex #7170 (LlamaIndex contributors, 2023) | user typo "occrra"→"occcra" propagates into hallucinated tool name; dispatch crashes |
| QueryPara | LlamaIndex #16757 (LlamaIndex contributors, 2024) | query "summarise the document" routes to *vector_doc* (search) instead of *list_doc*; fixed by editing tool descriptions |
| ToolPara | LlamaIndex #16757; (Faghih et al., 2025) | paraphrasing tool descriptions changes selection; (Faghih et al., 2025) measure 10× usage variance |
| ParamPara | grafana/loki-mcp #27 (Grafana Loki MCP contributors, 2025) | parameter description lists "1h ago" default; agents send the literal string but parser only accepts *-1h*/RFC3339/*now* |
| **Action perturbations** (6 types) | | |
| Dup-* | openai-agents-python #1167 (openai-agents-python contributors, 2025) | two MCP servers register the same tool name; the SDK hangs indefinitely |
| RedunTool | tau-bench #39 (Sierra Research, 2025) | "direct flights" tool description vs. one-stop-flights implementation; agent picks the wrong sibling |
| **Reward perturbations** (6 types) | | |
| MisDesc | netbox-mcp-server #79 (NetBox Labs, 2025) | misleading filter description silently returns all records instead of the filtered subset |
| TimeDesc | tau-bench #39 (Sierra Research, 2025); (Faghih et al., 2025) | description-vs-implementation mismatch (*search_onestop_flight* described as "direct flights", a speed/time implication); we cite (Faghih et al., 2025) for the general description-wording mechanism that TimeDesc stress-tests |
| MisDesc-N, TimeDesc-N | crystaldba/postgres-mcp #157 (crystaldba contributors, 2025) | nine sibling tools with vague descriptions; LLM picks the wrong sibling; fixed by "do not use when…" clauses |
| MisDesc-Abbr | netbox-mcp #79; (Faghih et al., 2025) | abbreviation amplifies the MisDesc failure; surface-naming bias is the documented mechanism |
| TimeDesc-Abbr | semantic-kernel #13690 (Microsoft Semantic Kernel contributors, 2025); (Faghih et al., 2025) | tool integrity verification gap (MCP servers can swap tools mid-session); we synthesize the time + abbreviation case as a stress-test of the description-wording mechanism in (Faghih et al., 2025) |
| **Transition perturbations** (6 types) | | |
| Timeout | langchain #35597 (LangChain contributors, 2025c) | default *request_timeout=None* causes agents to hang indefinitely |
| RateLim | openai-python #2699 (OpenAI Python contributors, 2025) | undocumented HTTP 429 asymmetry across endpoints |
| AuthErr | langchain #29596 (LangChain contributors, 2025a) | missing Authorization header causes silent 401 |
| 5xxErr | Gorilla #839 (Gorilla contributors, 2025) | vLLM disconnects mid-inference during BFCL evaluation |
| Malform | langchain #34746 (LangChain contributors, 2025b) | Ollama returns malformed JSON; tool call silently dropped |
| SchemaD | langchain #36032 (LangChain contributors, 2025d) | *anyOf* schema crashes Ollama after a tool-definition update |

**Audit procedure.** For every issue cited in Table 5 we re-fetched the issue body during the v2 revision and confirmed (a) the URL still resolves, (b) the issue content matches the perturbation it grounds, (c) the source repository is part of the tool-calling / agent / MCP / function-calling ecosystem (no vision-only or general-LLM bugs).

**Demoted entries.** Two issues that the team initially labeled "strong" did not survive re-verification and were replaced:

- *apple-calendar-mcp #270* (originally cited under ToolPara) actually documents an *eval-scorer bug* and missing alternative tool documentation, not description-paraphrase steering. Replaced with LlamaIndex #16757 (the "summarise the document" routing case) plus the academic measurement in Faghih et al. (Faghih et al., 2025), which directly quantifies how tool-description wording can change selection by up to $10\times$.
- *autogen #6935* (originally cited under MisDesc-Abbr) actually documents a problem with the OpenAI Responses-API JSON wrapper (a function dictionary nesting issue), not abbreviation-driven misselection. Replaced with netbox-mcp-server #79 (mechanism: misleading description on a filter field) plus Faghih et al. (Faghih et al., 2025).

**Caveat: types with softer GitHub-issue evidence.** Four types do not have a single GitHub issue that exactly matches our perturbation construction; we back them with academic measurement and / or related-mechanism issues, and acknowledge the synthesis here:

- **ToolPara**: LlamaIndex #16757 documents description-rewrite changing tool selection, but the academic measurement in Faghih et al. (Faghih et al., 2025) ($10\times$ usage variance under description paraphrase) is the more direct quantitative evidence.
- **MisDesc-Abbr**: netbox-mcp #79 documents misleading description; abbreviation amplifies that mechanism. We synthesize the abbreviation case rather than citing a separate abbreviation-specific production issue.
- **TimeDesc**: tau-bench #39 documents *search_onestop_flight* described as "direct flights" (a speed/time-implication mismatch), which is closely related but not specifically about *response-time annotations*. We cite Faghih et al. (Faghih et al., 2025) for the general description-wording mechanism that TimeDesc stress-tests.
- **TimeDesc-Abbr**: semantic-kernel #13690 documents a broader tool-integrity verification gap rather than a time + abbreviation case specifically. The TimeDesc-Abbr construction combines the description-wording mechanism (Faghih et al.) with the abbreviation pattern from MisDesc-Abbr.

The other 18 types have direct, single-issue grounding; see Table 5.

**Source-ecosystem check.** The 13 distinct repositories cited in §4.2 are all tool-call / agent / MCP / function-calling / benchmark projects: LlamaIndex (LLM agent framework, 2 issues), LangChain (LLM agent framework, 4 issues), Microsoft Semantic Kernel (agent framework), Microsoft AutoGen (initially cited; demoted), OpenAI Agents Python (agent SDK), apple-calendar-mcp / grafana-loki-mcp / netbox-mcp / crystaldba-postgres-mcp (4 MCP tool servers), ShishirPatil/gorilla (BFCL benchmark), sierra-research/tau-bench (tool-use benchmark), openai-python (OpenAI SDK used by tool agents). No vision-model or general-LLM repositories appear in the citation list.

## F. Generation prompts

Five of the perturbations require LLM assistance (the other 17 are rule-based). The exact prompts handed to gpt-5-mini are reproduced below; the variable in `[brackets]` is substituted at runtime with the corresponding field from each clean source sample.

**Realistic typos (Typo).** *"Add realistic typing errors to the following query, simulating natural human typos. [query] Requirements: add 2–4 realistic typos that humans commonly make when typing quickly; include common typo types (adjacent key hits e→r, character swaps 'teh'→'the', missing letters, doubled letters, common misspellings); DO NOT change any numbers, dates, proper nouns, or technical terms; DO NOT change the meaning or intent of the query; output the perturbed query only."*

**Query paraphrase (QueryPara).** *"Paraphrase the following user query while preserving its exact meaning and intent. [query] Requirements: use different wording but keep the same semantic meaning; DO NOT change any locations, person*

*names, numbers, dates, or specific entities; maintain all technical terms and important details; output the paraphrased query only."*

**Tool-description paraphrase (ToolPara).** *"Paraphrase the following tool/function description while preserving its exact meaning. Tool name: [tool_name]. Original description: [description]. Requirements: use different wording but keep the same semantic meaning; maintain all technical details and constraints; keep similar length ($\pm 20\%$); output ONLY the paraphrased description (no explanation)."*

**Parameter-description paraphrase (ParamPara).** *"Paraphrase the following API parameter description while preserving its exact meaning. Parameter name: [param_name]. Parameter type: [param_type]. Original description: [description]. Requirements: use different wording but keep the same semantic meaning; maintain type constraints and valid values; keep similar length; output ONLY the paraphrased description (no explanation)."*

**Redundant similar tool (RedunTool).** *"You are an API designer. Given the following existing tool, generate [num_tools] NEW tools that are semantically related but serve DIFFERENT purposes. Existing tool: [existing_tool]. Requirements: the new tools should be plausible extensions that could exist alongside the existing tool; they should NOT duplicate existing functionality; each tool needs a descriptive name following the same naming convention, a clear description, and appropriately typed parameters. Output as a JSON array of tool dicts."*

The audit in Appendix H verifies that the four paraphrase prompts produce semantically-equivalent outputs in the released dataset.

## G. Transition error strings

Transition perturbations are injected at runtime by the eval harness (*scripts/run_eval.py –mode transition –transition-type ¡T¿*). The first inference pass receives the clean tool list; if the model emits a tool call, the harness injects a fixed error string as the (simulated) tool response, then runs a second inference pass. We deliberately use canonical error strings rather than live API errors to keep evaluation reproducible across reruns. The six strings are listed below verbatim; their wording is drawn from the cited GitHub issues in Table 5.

- **timeout**: *"Tool execution timed out after the configured request timeout. The remote endpoint did not respond within the allotted time."*
- **rate_limit**: *"HTTP 429 Too Many Requests. The provider rejected the call because the per-minute rate limit has been exceeded."*
- **auth_error**: *"HTTP 401 Unauthorized. The provider rejected the call because the supplied credentials are invalid or expired."*
- **server_error**: *"HTTP 500 Internal Server Error. The remote endpoint failed to handle the request."*
- **malformed_response**: *"Malformed response from tool execution: the body could not be parsed as JSON."*
- **schema_drift**: *"Schema validation failed: the response did not match the tool's declared output schema (extra/missing fields)."*

## H. Paraphrase quality audit

**Method.** For each of the four LLM-generated types (Typo, QueryPara, ToolPara, ParamPara) we randomly sample $n = 50$ pairs of (clean, perturbed) text from the released dataset. A judge LLM (gpt-5.4) reads each pair and scores the perturbation on a 1–5 semantic-equivalence scale: 5 = identical intent and information; 4 = same intent, minor wording loss that would not change the correct tool call; 3 = same intent but a parameter or constraint became ambiguous; 2 = intent shifted, the GT tool call may no longer be unambiguous; 1 = intent broken. The judge runs at its default sampling (gpt-5.4 is a reasoning model that does not accept arbitrary temperature). Per-pair score variance is small relative to the type-mean standard error ($\approx 0.4/\sqrt{50} \approx 0.06$ on a 1–5 scale). Of 200 sampled pairs, 196 had non-empty perturbed text under the per-type field extraction (4 ParamPara samples skipped because the source benchmark stored no parameter descriptions for that sample's tools).

**Results.** All four types exceed mean 4.7 on the 1–5 scale, with no pair scored at $\leq 2$. The perturbations preserve intent across the audited sample.

*Table 6.* Paraphrase quality audit. For each LLM-generated subtype we audit $n$ random (clean, perturbed) pairs with a fixed-temperature judge and report mean / median / standard deviation of the 1–5 semantic-equivalence score, plus the fraction of pairs scored $\leq 2$ (intent shifted or broken). Lower fractions are better.

| Subtype | $n$ | Mean | Median | Std | Fraction $\leq 2$ |
|---|---|---|---|---|---|
| Typo | 50 | 4.94 | 5.0 | 0.31 | 0.00% |
| QueryPara | 50 | 5.00 | 5.0 | 0.00 | 0.00% |
| ToolPara | 50 | 4.98 | 5.0 | 0.14 | 0.00% |
| ParamPara | 46 | 4.72 | 5.0 | 0.58 | 0.00% |

**Lowest-scoring examples.** The five lowest-scoring pairs in each type are listed below. For Typo, QueryPara, and ToolPara even the lowest scores fall in the 3–5 range, indicating only minor ambiguity introduced by the perturbation. ParamPara is the most demanding type because parameter descriptions are short and dense, so individual word choices (e.g., "standard deviation" rewritten as "variance measure") can shift meaning; even so, no pair was rated $\leq 2$.

**Typo — lowest-scoring 5 audited pairs. Score 3** *(sample id: `apibank__level1_375`)*

*Clean:* "**Dialogue Records History** ¡user¿Can you help me register an appointment at the hospital?¡/user¿ ¡response¿Of course, I can do that for you. Please provide me with the patient name, appointment date…"

*Perturbed:* "**Dialogue Records History** ¡user¿Can you help me register an appointment at the hospital?¡/user¿ ¡response¿Of course, I can do that for you. Please provide me with the patient name, appointment date…"

*Judge:* The intent is the same, but the patient and doctor names are altered by typos ('Jonh' and 'Dr. Smmith'), making key tool-call parameters ambiguous.

**Score 4** *(sample id: `tooleyes__Turn 1: Can you help me by generating four UUIDs for the new virtual machines in our network?`)*

*Clean:* "Can you help me by generating four UUIDs for the new virtual machines in our network?…"

*Perturbed:* "Can you halp me by generating four UUIDs for teh new virtual mchine in our network?…"

*Judge:* The typos do not change the intent to generate four UUIDs, though 'virtual mchine' introduces a minor singular/plural wording loss that would not affect the same tool call.

**Score 5** *(sample id: `apibank__level1_200`)*

*Clean:* "**Dialogue Records History** ¡user¿Can you help me record my health history?¡/user¿ ¡response¿Sure, I can help you with that. What is your user ID and health data you want to record?¡/response¿ ¡user…"

*Perturbed:* "**Dialogue Records History** ¡user¿Can you help me record my health history?¡/user¿ ¡response¿Sure, I can help you with that. What is your user ID and health data you want to record?¡/response¿ ¡user…"

*Judge:* The typos in 'recored' and 'presure' do not change the intent, entities, timestamp, or the correct single tool call to record blood pressure and heart rate for user 12345.

**Score 5** *(sample id: `tooleyes__Turn 1: Please tell me the definition of the word 'hello'.`)*

*Clean:* "Please tell me the definition of the word 'hello'.…"

*Perturbed:* "Pleas tell me teh definiton of teh word 'hello'.…"

*Judge:* The typos do not alter the request to define the word 'hello', so the same tool call would be made.

**Score 5** *(sample id: `bfcl_v3__BFCL_v3_multiple__multiple_61`)*

*Clean:* "Find a Landscape Architect who is experienced 5 years in small space garden design in Portland…"

*Perturbed:* "Fidn a Lndscape Architect who is experinced 5 yeras in small space garden desgn in Portland…"

*Judge:* Despite multiple typos, the request clearly preserves the same intent and all key constraints: Landscape Architect, 5 years experience, small space garden design, and Portland.

**QueryPara — lowest-scoring 5 audited pairs. Score 5** *(sample id: `apibank__level1_44`)*

*Clean:* "**Dialogue Records History** ¡user¿Can you help me reschedule my appointment?¡/user¿ ¡response¿Of course. Please provide me with your appointment ID, the new appointment date, and the new doctor's nam…"

*Perturbed:* "**Dialogue Records History** ¡user¿Can you help me reschedule my appointment?¡/user¿ ¡response¿Of course. Please provide me with your appointment ID, the new appointment date, and the new doctor's nam…"

*Judge:* The perturbation preserves the exact rescheduling intent and all required parameters: appointment ID 90123456, new date October 20th, and doctor Dr. Johnson.

**Score 5** *(sample id: `tooleyes__Turn 1: What day is it today?`)*

*Clean:* "What day is it today?…"

*Perturbed:* "What is the date today?…"

*Judge:* Both ask for the current calendar information and would require the same date/time lookup tool call.

**Score 5** *(sample id: `apibank__level1_251`)*

*Clean:* "**Dialogue Records History** ¡user¿Can you help me open a bank account?¡/user¿ ¡response¿Sure. To open a bank account, I'll need your account number, password, and name.¡/response¿ ¡user¿My account n…"

*Perturbed:* "**Dialogue Records History** ¡user¿Can you help me open a bank account?¡/user¿ ¡response¿Sure. To open a bank account, I'll need your account number, password, and name.¡/response¿ ¡user¿My account I…"

*Judge:* Replacing 'account number' with 'account ID' preserves the same intent and all needed parameters for the same single bank-account creation tool call.

**Score 5** *(sample id: `tooleyes__Turn 1: Search for e-prints related to the topic of genetic algorithms using the arXiv API. Display the first 3 results.`)*

*Clean:* "Search for e-prints related to the topic of genetic algorithms using the arXiv API. Display the first 3 results.…"

*Perturbed:* "Look for e-prints associated with genetic algorithms through the arXiv API. Show the top 3 results.…"

*Judge:* The perturbation preserves the same search topic, arXiv API usage, and constraint to display the first/top 3 results, so the same tool call applies.

**Score 5** *(sample id: `apibank__level1_184`)*

*Clean:* "**Dialogue Records History** ¡user¿Hi, can you help me add a new schedule for my meeting tomorrow at 2pm? Today is 2021-09-27¡/user¿ ¡response¿Sure, I can help you with that. What's the content of the…"

*Perturbed:* "**Dialogue Records History** ¡user¿Hi, can you help me add a new schedule for my meeting tomorrow at 2pm? Today is 2021-09-27¡/user¿ ¡response¿Sure, I can help you with that. What's the content of the…"

*Judge:* The perturbation only changes the phrasing of the credential sentence while preserving the same username, password, intent, and all scheduling details needed for the identical tool call.

**ToolPara — lowest-scoring 5 audited pairs.** **Score 4** *(sample id: `tooleyes__Turn 1: Get me all latest Net Asset Value.`)*

*Clean:* "["Fetch Latest NAV. These APIs provide latest NAV information of all mutual funds in India from Association of Mutual Funds of India (AMFI).", "Fetch Historical NAV. These APIs provide latest NAV info…"

*Perturbed:* "["Retrieve Most Recent NAV. These APIs supply the current NAV data for all mutual funds in India as provided by the Association of Mutual Funds of India (AMFI).", "Retrieve Historical NAV. These APIs …"

*Judge:* The perturbation mostly preserves the same tool-use intents, but 'Fetch All Scheme Types' was changed to 'Retrieve All Scheme Categories' and 'All Mutual Fund Families' to 'Comprehensive Data on Mutual Fund Families,' introducing minor ambi

**Score 5** *(sample id: `tooleyes__Turn 1: Make a guess about Jane's gender.`)*

*Clean:* "["Predicts the ages of one or more people given their names.", "Predicts the genders of one or more people given their names.", "Predicts the nationalities of one or more people given their names.", "…"

*Perturbed:* "["Estimates the ages of individuals based on their names, whether it be for one person or several.", "Estimates the genders of individuals based on their names, whether it's one person or several.", "…"

*Judge:* The perturbation preserves the same capabilities and constraints, with only synonymous wording changes that do not affect the intended tool use.

**Score 5** *(sample id: `apibank__level2_11`)*

*Clean:* "["This API gets the current date.", "** Recall relevant context and analyze the current user goal."]..."

*Perturbed:* "["This API retrieves the current date.", "** Recall relevant context and analyze the current user goal."]..."

*Judge:* 'Gets' to 'retrieves' is a synonymous rewording that preserves the exact intent and tool call.

**Score 5** *(sample id: `apibank__level1_103`)*

*Clean:* "["This is an API for opening a bank account for a user, given the account, password and name.", "This API queries the stock price of a given stock code and date.", "This API queries the balance of a g..."

*Perturbed:* "["This API opens a user's bank account given the account, password and name.", "This API retrieves the stock price for a specified stock code on a particular date.", "This API retrieves the account ba..."

*Judge:* All API descriptions preserve the same intent, parameters, and constraints with only synonymous rewording that would not change any tool call.

**Score 5** *(sample id: `toolalpaca__Cataas__inst2`)*

*Clean:* "["Get random cat", "Get cat by id", "Get random cat by tag", "Get random cat saying text", "Will return all cats", "Will return all tags", "Count how many cats"]..."

*Perturbed:* "["Retrieve a random cat \{}nParameters: {\{}"type\{}": \{}"string. Specify the type of cat.\{}", \{}"width\{}": \{}"string. Preferred width of the cat image.\{}", \{}"height\{}": \{}"string. Preferred height of the cat imag..."

*Judge:* All seven descriptions preserve the same underlying operations and required distinctions between endpoints, with added parameter detail that does not alter which tool should be called.

**ParamPara — lowest-scoring 5 audited pairs.** **Score 3** *(sample id: `bfcl_v3__BFCL_v3_multiple__multiple_36`)*

*Clean:* "kinematics.calculate_acceleration.initial_speed: The initial speed of the object. kinematics.calculate_acceleration.final_speed: The final speed of the object. kinematics.calculate_acceleration.time: ..."

*Perturbed:* "kinematics.calculate_acceleration.initial_speed: The object's starting velocity. kinematics.calculate_acceleration.final_speed: The object's speed at the end. kinematics.calculate_acceleration.time: T..."

*Judge:* Most parameter meanings are preserved, but changing acceleration.time to the time to achieve its maximum speed introduces ambiguity and a semantic shift from simply reaching the final speed.

**Score 3** *(sample id: `bfcl_v3__BFCL_v3_multiple__multiple_191`)*

*Clean:* "random.normalvariate.mu: Mean of the normal distribution. random.normalvariate.sigma: Standard deviation of the normal distribution. get_personality_traits.type: The personality type. get_personality_..."

*Perturbed:* "random.normalvariate.mu: Average value of the normal distribution. random.normalvariate.sigma: Variance measure of the Gaussian distribution. get_personality_traits.type: The classification of persona..."

*Judge:* Most parameter meanings are preserved, but changing random.normalvariate.sigma from standard deviation to a vague 'variance measure' and book_hotel.location from location to 'address' introduces ambiguity that could affect the exact tool ca

**Score 3** *(sample id: `rotbench__Turn 1: To which country does the city of Madrid belong and what is its population?`)*

*Clean:* "search_country.query: The name or IATA code of the country. search_country.key: Determine whether query is a name (default) or an IATA code. It must be either "name" or "code". search_city.query: The ..."

*Perturbed:* "search_country.query: The designation or IATA code for the nation. search_country.key: Specify if the query represents a name (default) or an IATA code. It should be either "name" or "code". search_ci..."

*Judge:* Most parameter meanings are preserved, but search_city.query changes from specifically a city to the broader

'location' and search_routes.max_transfers is altered from 'max number of directs' to 'direct transfers allowed,' introducing ambig

**Score 4** *(sample id: `toolalpaca__Cataas__inst1`)*

*Clean:* "getRandomCat.type: Filter by cat type getRandomCat.width: Desired width of the cat image getRandomCat.height: Desired height of the cat image getRandomCat.html: Include HTML code getRandomCat.json: In..."

*Perturbed:* "getRandomCat.type: Restrict results based on category type getRandomCat.width: Preferred width of the cat image getRandomCat.height: Preferred height for the cat image getRandomCat.html: Incorporate H..."

*Judge:* The perturbation largely preserves the same tool-call semantics, but terms like 'category type,' 'label,' and 'spoken by the cat' introduce minor ambiguity without changing the likely correct parameters.

**Score 4** *(sample id: `rotbench__Turn 1: Could tell me 3 facts of cats and dogs?`)*

*Clean:* "cat_breed.limit: Limit the amount of results returned. cat_facts.max_length: The maximum length of returned fact. cat_facts.limit: Limit the amount of results returned. random_dog_image.limit: Limit t..."

*Perturbed:* "cat_breed.limit: Restrict the number of results provided. cat_facts.max_length: The upper limit for the length of the fact provided in the response. cat_facts.limit: Restrict the number of results tha..."

*Judge:* The perturbation preserves the same tool parameters and constraints overall, with only minor wording changes like 'subcategory/variant' for 'sub-breed' that do not alter the intended tool call.

## I. Per-benchmark scoring rules

Scoring is performed by *scripts/eval_results.py* which dispatches each prediction to its source benchmark's scorer. Every scorer takes as input the parsed *tool_calls* list (a list of {`name, parameters`} dicts) and the per-sample *golden_answers*. The parsing step that produces *tool_calls* from the model's *raw_output* is intentionally format-tolerant; the scoring step is then strict.

**Tool-call parser variants.** The format-tolerant parser (*parse_tool_calls()* in *scripts/run_eval.py*) tries the following variants, in dispatch order keyed on the source benchmark:

- *BFCL canonical*: `[func(a=1)]` (Python AST). Also accepts the colon-separator variant `[func(a:1)]` (Qwen2.5-1.5B ToolRL) and the keyword-list variant `[func_name="X", params={...}]` (SFT-Clean).
- *API-Bank XML*: `<tool_call>{json}</tool_call>`, plus the variants `<tool>X</tool>\n{json}`, `<tool name="X" parameters={...}/>`, `<toolcall tool="X">`, `<tool_call tool_name="X">` (Qwen2.5-1.5B).
- *ReAct*: `Action: X\nAction Input: {json}`; also accepts `ActionCode:` alias (SFT-Robust).
- *ToolAlpaca ad-hoc*: `func_name: {json}` (Qwen3.5), `Function: X\nParameters: Y` (Qwen2.5-3B), inline query-string `` `fn?p1=v1&p2=v2` `` (Llama-3.2-Instruct).
- *JSON blob*: a JSON object with name aliases (*name—function—tool—func_name—tool_name—action*) and parameter aliases (*parameters—arguments—params—args—action_input*).
- *Bare Python call*: a fallback used by RoTBench when TL-CodeLLaMA emits raw Python.

For each benchmark, the parser tries variants in the order shown until one returns a non-empty list; if every variant fails the prediction is recorded as *tool_calls=[]* and contributes to the *omitted tool call* error mode (Appendix L).

**Per-benchmark scorers.**

- *BFCL V3 (ST)*: AST-based exact match on tool name and required parameters; lenient on missing optional parameters (matches the upstream BFCL scorer).
- *API-Bank*: exact match on tool name; parameter values are compared using API-Bank's per-parameter type rules (case-insensitive for strings, normalized for numerics).
- *RoTBench*: ReAct exact match on *Action* name; *Action Input* is JSON-decoded and compared field-by-field.
- *ToolAlpaca*: most lenient — name match plus subset match on parameter values (because the source benchmark mixes ReAct, XML, and bare Python ground-truth formats inconsistently).

- *ToolEyes*: ReAct exact match on *Action* name; partial credit on multi-tool sequences.

The scorer never invokes an LLM judge; the per-benchmark rules above are the entire scoring surface.

## J. Per-perturbation accuracy and drop tables

The main paper's Table 2 reports per-POMDP-component summary numbers. This appendix gives the full breakdown: every (model, perturbation type) pair with both raw accuracy and drop-from-clean (in parentheses). Models are grouped into the same four families used in the main results table; horizontal lines separate the groups. We split the 22 perturbations into one sub-table per POMDP component to keep each table page-fittable.

*Table 7.* Per-perturbation accuracy under **Observation** perturbations (part 1/2). The first numeric column is Clean accuracy (the unperturbed baseline); each subsequent entry shows *accuracy on that perturbation* $\pm$ 95% bootstrap half-width, with the drop *from this model's own Clean* (also $\pm$ 95% half-width) in parentheses. A more negative parenthesised number means a larger sim-to-real gap (worse robustness). All values are accuracies in $[0, 1]$, matching Table 2. $B = 10{,}000$ percentile-bootstrap resamples on per-sample correctness scores. **Bold** = our trained checkpoints.

| | Clean | Typo | QueryPara |
|---|---|---|---|
| *Our 3B trained checkpoints* | | | |
| **ToolRL-DR-Full (ours)** | **0.643±0.065** | **0.603±0.070 (-0.040±0.095)** | **0.653±0.065 (+0.010±0.093)** |
| **ToolRL-DR-Mixed (ours)** | **0.658±0.065** | **0.573±0.068 (-0.085±0.095)** | **0.623±0.065 (-0.035±0.093)** |
| *Open-source RL tool models, 8B–14B* | | | |
| MUA-RL-8B | 0.678±0.065 | 0.603±0.068 (-0.075±0.093) | 0.653±0.065 (-0.025±0.093) |
| MUA-RL-14B | 0.628±0.065 | 0.563±0.070 (-0.065±0.098) | 0.623±0.068 (-0.005±0.095) |
| LoopTool-8B | 0.714±0.063 | 0.583±0.070 (-0.131±0.088) | 0.688±0.065 (-0.025±0.090) |
| *Open-source RL tool models, 1.5B–7B* | | | |
| ToolRL-Qwen2.5-1.5B | 0.704±0.063 | 0.628±0.068 (-0.075±0.090) | 0.668±0.065 (-0.035±0.090) |
| ToolRL-Qwen2.5-3B | 0.638±0.065 | 0.563±0.070 (-0.075±0.095) | 0.643±0.065 (+0.005±0.093) |
| ToolRL-Llama3.2-3B | 0.628±0.065 | 0.553±0.070 (-0.075±0.095) | 0.598±0.068 (-0.030±0.095) |
| TL-CodeLLaMA-2 | 0.653±0.065 | 0.578±0.070 (-0.075±0.095) | 0.628±0.068 (-0.025±0.095) |
| *Open-source RL tool models, 32B* | | | |
| MUA-RL-32B | 0.683±0.065 | 0.598±0.070 (-0.085±0.090) | 0.658±0.065 (-0.025±0.093) |
| LoopTool-32B | 0.779±0.058 | 0.673±0.065 (-0.106±0.088) | 0.764±0.060 (-0.015±0.083) |
| *Base / instruct counterparts* | | | |
| Qwen2.5-1.5B-Instruct | 0.573±0.070 | 0.518±0.070 (-0.055±0.096) | 0.523±0.070 (-0.050±0.095) |
| Qwen2.5-3B-Instruct | 0.608±0.068 | 0.558±0.070 (-0.050±0.098) | 0.593±0.070 (-0.015±0.095) |
| Llama-3.2-3B-Instruct | 0.518±0.070 | 0.487±0.070 (-0.030±0.095) | 0.503±0.070 (-0.015±0.098) |
| Qwen3-8B | 0.673±0.065 | 0.588±0.068 (-0.085±0.095) | 0.668±0.065 (-0.005±0.093) |
| Qwen3-14B | 0.749±0.060 | 0.643±0.065 (-0.106±0.090) | 0.729±0.060 (-0.020±0.085) |
| Qwen3-32B | 0.754±0.058 | 0.653±0.065 (-0.101±0.088) | 0.749±0.060 (-0.005±0.085) |
| *Frontier (open weights, reasoning / general)* | | | |
| Qwen3.5-9B | 0.709±0.063 | 0.613±0.068 (-0.095±0.090) | 0.678±0.065 (-0.030±0.090) |
| DeepSeek-R1-Distill-14B | 0.618±0.070 | 0.568±0.068 (-0.050±0.096) | 0.618±0.068 (+0.000±0.095) |
| *Closed-source frontier* | | | |
| o4-mini (OpenAI) | 0.709±0.063 | 0.618±0.068 (-0.090±0.090) | 0.623±0.068 (-0.085±0.093) |
| *ToolRL SFT version* | | | |
| SFT-Clean-4k-3B | 0.598±0.070 | 0.533±0.070 (-0.065±0.095) | 0.588±0.070 (-0.010±0.095) |

*Table 8.* Per-perturbation accuracy under **Observation** perturbations (part 2/2). The first numeric column is Clean accuracy (the unperturbed baseline); each subsequent entry shows *accuracy on that perturbation* $\pm$ 95% bootstrap half-width, with the drop *from this model's own Clean* (also $\pm$ 95% half-width) in parentheses. A more negative parenthesised number means a larger sim-to-real gap (worse robustness). All values are accuracies in $[0, 1]$, matching Table 2. $B = 10{,}000$ percentile-bootstrap resamples on per-sample correctness scores. **Bold** = our trained checkpoints.

| | Clean | ToolPara | ParamPara |
|---|---|---|---|
| *Our 3B trained checkpoints* | | | |
| **ToolRL-DR-Full (ours)** | **0.643±0.065** | **0.643±0.065 (+0.000±0.095)** | **0.638±0.065 (-0.005±0.093)** |
| **ToolRL-DR-Mixed (ours)** | **0.658±0.065** | **0.633±0.068 (-0.025±0.090)** | **0.608±0.065 (-0.050±0.095)** |
| *Open-source RL tool models, 8B–14B* | | | |
| MUA-RL-8B | 0.678±0.065 | 0.693±0.065 (+0.015±0.090) | 0.673±0.065 (-0.005±0.091) |
| MUA-RL-14B | 0.628±0.070 | 0.643±0.065 (+0.015±0.095) | 0.638±0.068 (+0.010±0.095) |
| LoopTool-8B | 0.714±0.063 | 0.698±0.065 (-0.015±0.088) | 0.688±0.065 (-0.025±0.090) |
| *Open-source RL tool models, 1.5B–7B* | | | |
| ToolRL-Qwen2.5-1.5B | 0.704±0.060 | 0.683±0.065 (-0.020±0.090) | 0.704±0.063 (+0.000±0.090) |
| ToolRL-Qwen2.5-3B | 0.638±0.065 | 0.638±0.065 (+0.000±0.095) | 0.628±0.068 (-0.010±0.095) |
| ToolRL-Llama3.2-3B | 0.628±0.068 | 0.623±0.065 (-0.005±0.095) | 0.638±0.068 (+0.010±0.095) |
| TL-CodeLLaMA-2 | 0.653±0.065 | 0.668±0.065 (+0.015±0.093) | 0.643±0.068 (-0.010±0.095) |
| *Open-source RL tool models, 32B* | | | |
| MUA-RL-32B | 0.683±0.065 | 0.688±0.065 (+0.005±0.093) | 0.678±0.065 (-0.005±0.091) |
| LoopTool-32B | 0.779±0.058 | 0.764±0.058 (-0.015±0.080) | 0.779±0.058 (+0.000±0.080) |
| *Base / instruct counterparts* | | | |
| Qwen2.5-1.5B-Instruct | 0.573±0.070 | 0.528±0.068 (-0.045±0.095) | 0.563±0.070 (-0.010±0.098) |
| Qwen2.5-3B-Instruct | 0.608±0.068 | 0.623±0.068 (+0.015±0.095) | 0.603±0.068 (-0.005±0.095) |
| Llama-3.2-3B-Instruct | 0.518±0.070 | 0.523±0.070 (+0.005±0.096) | 0.523±0.070 (+0.005±0.098) |
| Qwen3-8B | 0.673±0.065 | 0.688±0.065 (+0.015±0.090) | 0.673±0.065 (+0.000±0.093) |
| Qwen3-14B | 0.749±0.060 | 0.729±0.060 (-0.020±0.085) | 0.739±0.060 (-0.010±0.085) |
| Qwen3-32B | 0.754±0.060 | 0.749±0.060 (-0.005±0.085) | 0.769±0.060 (+0.015±0.083) |
| *Frontier (open weights, reasoning / general)* | | | |
| Qwen3.5-9B | 0.709±0.063 | 0.698±0.065 (-0.010±0.090) | 0.683±0.065 (-0.025±0.090) |
| DeepSeek-R1-Distill-14B | 0.618±0.065 | 0.613±0.068 (-0.005±0.095) | 0.623±0.065 (+0.005±0.095) |
| *Closed-source frontier* | | | |
| o4-mini (OpenAI) | 0.709±0.063 | 0.719±0.060 (+0.010±0.090) | 0.678±0.065 (-0.030±0.090) |
| *ToolRL SFT version* | | | |
| SFT-Clean-4k-3B | 0.598±0.068 | 0.598±0.065 (+0.000±0.095) | 0.598±0.068 (+0.000±0.095) |

*Table 9.* Per-perturbation accuracy under **Action** perturbations (part 1/2). The first numeric column is Clean accuracy (the unperturbed baseline); each subsequent entry shows *accuracy on that perturbation* ± 95% bootstrap half-width, with the drop *from this model's own Clean* (also ± 95% half-width) in parentheses. A more negative parenthesised number means a larger sim-to-real gap (worse robustness). All values are accuracies in $[0, 1]$, matching Table 2. $B = 10{,}000$ percentile-bootstrap resamples on per-sample correctness scores. **Bold** = our trained checkpoints.

| | Clean | Dup-NoDesc | Dup-Desc | Dup-WrongP |
|---|---|---|---|---|
| *Our 3B trained checkpoints* | | | | |
| **ToolRL-DR-Full (ours)** | **0.643±0.065** | **0.444±0.083 (-0.200±0.107)** | **0.472±0.087 (-0.171±0.110)** | **0.500±0.087 (-0.143±0.108)** |
| **ToolRL-DR-Mixed (ours)** | **0.658±0.065** | **0.436±0.083 (-0.222±0.108)** | **0.441±0.087 (-0.217±0.108)** | **0.500±0.087 (-0.158±0.109)** |
| *Open-source RL tool models, 8B–14B* | | | | |
| MUA-RL-8B | 0.678±0.065 | 0.504±0.083 (-0.175±0.107) | 0.512±0.087 (-0.167±0.108) | 0.500±0.087 (-0.178±0.110) |
| MUA-RL-14B | 0.628±0.068 | 0.504±0.083 (-0.124±0.109) | 0.504±0.087 (-0.124±0.111) | 0.516±0.087 (-0.112±0.110) |
| LoopTool-8B | 0.714±0.063 | 0.526±0.083 (-0.187±0.108) | 0.520±0.087 (-0.194±0.107) | 0.532±0.087 (-0.182±0.108) |
| *Open-source RL tool models, 1.5B–7B* | | | | |
| ToolRL-Qwen2.5-1.5B | 0.704±0.063 | 0.481±0.083 (-0.222±0.105) | 0.465±0.087 (-0.239±0.107) | 0.563±0.087 (-0.140±0.108) |
| ToolRL-Qwen2.5-3B | 0.638±0.065 | 0.451±0.083 (-0.187±0.105) | 0.488±0.087 (-0.150±0.108) | 0.524±0.087 (-0.114±0.111) |
| ToolRL-Llama3.2-3B | 0.628±0.065 | 0.406±0.086 (-0.222±0.105) | 0.441±0.087 (-0.187±0.110) | 0.437±0.087 (-0.192±0.109) |
| TL-CodeLLaMA-2 | 0.653±0.065 | 0.451±0.083 (-0.202±0.107) | 0.441±0.087 (-0.212±0.109) | 0.500±0.087 (-0.153±0.110) |
| *Open-source RL tool models, 32B* | | | | |
| MUA-RL-32B | 0.683±0.065 | 0.654±0.083 (-0.029±0.102) | 0.740±0.075 (+0.057±0.097) | 0.738±0.079 (+0.055±0.100) |
| LoopTool-32B | 0.779±0.058 | 0.594±0.083 (-0.185±0.099) | 0.622±0.083 (-0.157±0.101) | 0.698±0.079 (-0.080±0.098) |
| *Base / instruct counterparts* | | | | |
| Qwen2.5-1.5B-Instruct | 0.573±0.070 | 0.406±0.083 (-0.167±0.108) | 0.307±0.083 (-0.266±0.106) | 0.389±0.083 (-0.184±0.110) |
| Qwen2.5-3B-Instruct | 0.608±0.065 | 0.383±0.083 (-0.225±0.108) | 0.433±0.087 (-0.175±0.110) | 0.397±0.087 (-0.211±0.110) |
| Llama-3.2-3B-Instruct | 0.518±0.068 | 0.293±0.075 (-0.224±0.104) | 0.291±0.079 (-0.226±0.106) | 0.333±0.083 (-0.184±0.107) |
| Qwen3-8B | 0.673±0.065 | 0.474±0.083 (-0.200±0.105) | 0.472±0.087 (-0.201±0.110) | 0.484±0.087 (-0.189±0.108) |
| Qwen3-14B | 0.749±0.060 | 0.586±0.083 (-0.162±0.100) | 0.630±0.083 (-0.119±0.101) | 0.571±0.087 (-0.177±0.106) |
| Qwen3-32B | 0.754±0.060 | 0.594±0.083 (-0.160±0.103) | 0.646±0.083 (-0.108±0.102) | 0.643±0.083 (-0.111±0.102) |
| *Frontier (open weights, reasoning / general)* | | | | |
| Qwen3.5-9B | 0.709±0.063 | 0.519±0.083 (-0.190±0.107) | 0.528±0.087 (-0.181±0.107) | 0.532±0.087 (-0.177±0.110) |
| DeepSeek-R1-Distill-14B | 0.618±0.065 | 0.414±0.083 (-0.205±0.108) | 0.465±0.087 (-0.154±0.110) | 0.429±0.087 (-0.190±0.110) |
| *Closed-source frontier* | | | | |
| o4-mini (OpenAI) | 0.709±0.063 | 0.519±0.083 (-0.190±0.105) | 0.559±0.087 (-0.149±0.107) | 0.548±0.087 (-0.161±0.108) |
| *ToolRL SFT version* | | | | |
| SFT-Clean-4k-3B | 0.598±0.068 | 0.361±0.083 (-0.237±0.107) | 0.378±0.083 (-0.220±0.107) | 0.413±0.087 (-0.185±0.110) |

*Table 10.* Per-perturbation accuracy under **Action** perturbations (part 2/2). The first numeric column is Clean accuracy (the unperturbed baseline); each subsequent entry shows *accuracy on that perturbation* ± 95% bootstrap half-width, with the drop *from this model's own Clean* (also ± 95% half-width) in parentheses. A more negative parenthesised number means a larger sim-to-real gap (worse robustness). All values are accuracies in $[0, 1]$, matching Table 2. $B = 10{,}000$ percentile-bootstrap resamples on per-sample correctness scores. **Bold** = our trained checkpoints.

| | Clean | Dup-DescWP | Dup-SwapDP | RedunTool |
|---|---|---|---|---|
| *Our 3B trained checkpoints* | | | | |
| **ToolRL-DR-Full (ours)** | **0.643±0.065** | **0.472±0.088 (-0.171±0.110)** | **0.468±0.087 (-0.175±0.107)** | **0.578±0.068 (-0.065±0.095)** |
| **ToolRL-DR-Mixed (ours)** | **0.658±0.065** | **0.496±0.088 (-0.162±0.108)** | **0.476±0.087 (-0.182±0.109)** | **0.623±0.068 (-0.035±0.095)** |
| *Open-source RL tool models, 8B–14B* | | | | |
| MUA-RL-8B | 0.678±0.063 | 0.496±0.088 (-0.182±0.111) | 0.516±0.087 (-0.163±0.108) | 0.578±0.068 (-0.101±0.093) |
| MUA-RL-14B | 0.628±0.065 | 0.464±0.088 (-0.164±0.112) | 0.690±0.079 (+0.062±0.105) | 0.548±0.070 (-0.080±0.095) |
| LoopTool-8B | 0.714±0.065 | 0.528±0.088 (-0.186±0.108) | 0.516±0.087 (-0.198±0.109) | 0.593±0.068 (-0.121±0.093) |
| *Open-source RL tool models, 1.5B–7B* | | | | |
| ToolRL-Qwen2.5-1.5B | 0.704±0.063 | 0.528±0.088 (-0.176±0.106) | 0.516±0.087 (-0.188±0.108) | 0.593±0.068 (-0.111±0.093) |
| ToolRL-Qwen2.5-3B | 0.638±0.068 | 0.488±0.088 (-0.150±0.109) | 0.413±0.087 (-0.225±0.110) | 0.568±0.070 (-0.070±0.095) |
| ToolRL-Llama3.2-3B | 0.628±0.068 | 0.424±0.088 (-0.204±0.110) | 0.429±0.087 (-0.200±0.109) | 0.553±0.070 (-0.075±0.098) |
| TL-CodeLLaMA-2 | 0.653±0.068 | 0.456±0.088 (-0.197±0.112) | 0.468±0.087 (-0.185±0.110) | 0.563±0.070 (-0.090±0.095) |
| *Open-source RL tool models, 32B* | | | | |
| MUA-RL-32B | 0.683±0.065 | 0.720±0.080 (+0.037±0.103) | 0.730±0.079 (+0.047±0.101) | 0.623±0.065 (-0.060±0.090) |
| LoopTool-32B | 0.779±0.058 | 0.664±0.084 (-0.115±0.100) | 0.643±0.083 (-0.136±0.103) | 0.709±0.063 (-0.070±0.085) |
| *Base / instruct counterparts* | | | | |
| Qwen2.5-1.5B-Instruct | 0.573±0.070 | 0.280±0.080 (-0.293±0.104) | 0.333±0.079 (-0.240±0.107) | 0.467±0.070 (-0.106±0.098) |
| Qwen2.5-3B-Instruct | 0.608±0.068 | 0.320±0.084 (-0.288±0.106) | 0.333±0.083 (-0.275±0.106) | 0.538±0.070 (-0.070±0.095) |
| Llama-3.2-3B-Instruct | 0.518±0.070 | 0.272±0.080 (-0.246±0.106) | 0.341±0.083 (-0.176±0.107) | 0.452±0.070 (-0.065±0.098) |
| Qwen3-8B | 0.673±0.065 | 0.464±0.088 (-0.209±0.107) | 0.484±0.087 (-0.189±0.109) | 0.588±0.070 (-0.085±0.095) |
| Qwen3-14B | 0.749±0.060 | 0.576±0.088 (-0.173±0.106) | 0.619±0.087 (-0.130±0.103) | 0.633±0.065 (-0.116±0.090) |
| Qwen3-32B | 0.754±0.060 | 0.656±0.080 (-0.098±0.103) | 0.643±0.083 (-0.111±0.103) | 0.693±0.065 (-0.060±0.088) |
| *Frontier (open weights, reasoning / general)* | | | | |
| Qwen3.5-9B | 0.709±0.063 | 0.512±0.088 (-0.197±0.108) | 0.548±0.087 (-0.161±0.106) | 0.583±0.070 (-0.126±0.090) |
| DeepSeek-R1-Distill-14B | 0.618±0.065 | 0.408±0.088 (-0.210±0.110) | 0.484±0.087 (-0.134±0.110) | 0.518±0.070 (-0.101±0.098) |
| *Closed-source frontier* | | | | |
| o4-mini (OpenAI) | 0.709±0.063 | 0.552±0.088 (-0.157±0.107) | 0.524±0.087 (-0.185±0.108) | 0.588±0.068 (-0.121±0.090) |
| *ToolRL SFT version* | | | | |
| SFT-Clean-4k-3B | 0.598±0.068 | 0.344±0.084 (-0.254±0.106) | 0.389±0.083 (-0.209±0.110) | 0.533±0.068 (-0.065±0.098) |

*Table 11.* Per-perturbation accuracy under **Reward** perturbations (part 1/2). The first numeric column is Clean accuracy (the unperturbed baseline); each subsequent entry shows *accuracy on that perturbation* ± 95% bootstrap half-width, with the drop *from this model's own Clean* (also ± 95% half-width) in parentheses. A more negative parenthesised number means a larger sim-to-real gap (worse robustness). All values are accuracies in $[0, 1]$, matching Table 2. $B = 10{,}000$ percentile-bootstrap resamples on per-sample correctness scores. **Bold** = our trained checkpoints.

| | Clean | MisDesc | TimeDesc | MisDesc-N |
|---|---|---|---|---|
| *Our 3B trained checkpoints* | | | | |
| **ToolRL-DR-Full (ours)** | **0.643±0.065** | **0.461±0.098 (-0.182±0.118)** | **0.392±0.098 (-0.251±0.117)** | **0.451±0.098 (-0.192±0.116)** |
| **ToolRL-DR-Mixed (ours)** | **0.658±0.065** | **0.451±0.098 (-0.207±0.117)** | **0.343±0.088 (-0.315±0.113)** | **0.441±0.098 (-0.217±0.116)** |
| *Open-source RL tool models, 8B–14B* | | | | |
| MUA-RL-8B | 0.678±0.065 | 0.324±0.093 (-0.355±0.110) | 0.275±0.088 (-0.404±0.109) | 0.451±0.098 (-0.227±0.114) |
| MUA-RL-14B | 0.628±0.068 | 0.382±0.093 (-0.246±0.116) | 0.333±0.088 (-0.295±0.113) | 0.441±0.098 (-0.187±0.120) |
| LoopTool-8B | 0.714±0.063 | 0.373±0.098 (-0.341±0.114) | 0.304±0.088 (-0.410±0.108) | 0.480±0.098 (-0.233±0.116) |
| *Open-source RL tool models, 1.5B–7B* | | | | |
| ToolRL-Qwen2.5-1.5B | 0.704±0.063 | 0.265±0.083 (-0.439±0.107) | 0.245±0.083 (-0.458±0.106) | 0.402±0.098 (-0.302±0.116) |
| ToolRL-Qwen2.5-3B | 0.638±0.065 | 0.196±0.074 (-0.442±0.101) | 0.196±0.074 (-0.442±0.102) | 0.392±0.098 (-0.246±0.116) |
| ToolRL-Llama3.2-3B | 0.628±0.068 | 0.098±0.054 (-0.530±0.087) | 0.176±0.074 (-0.452±0.099) | 0.284±0.088 (-0.344±0.111) |
| TL-CodeLLaMA-2 | 0.653±0.065 | 0.196±0.074 (-0.457±0.100) | 0.294±0.088 (-0.359±0.109) | 0.353±0.093 (-0.300±0.114) |
| *Open-source RL tool models, 32B* | | | | |
| MUA-RL-32B | 0.683±0.065 | 0.353±0.088 (-0.330±0.114) | 0.333±0.088 (-0.350±0.111) | 0.422±0.098 (-0.262±0.114) |
| LoopTool-32B | 0.779±0.058 | 0.333±0.088 (-0.446±0.107) | 0.314±0.088 (-0.465±0.108) | 0.422±0.098 (-0.357±0.112) |
| *Base / instruct counterparts* | | | | |
| Qwen2.5-1.5B-Instruct | 0.573±0.070 | 0.225±0.078 (-0.347±0.106) | 0.196±0.074 (-0.377±0.100) | 0.275±0.088 (-0.298±0.111) |
| Qwen2.5-3B-Instruct | 0.608±0.065 | 0.206±0.078 (-0.402±0.103) | 0.147±0.064 (-0.461±0.097) | 0.324±0.088 (-0.285±0.114) |
| Llama-3.2-3B-Instruct | 0.518±0.070 | 0.137±0.064 (-0.380±0.095) | 0.108±0.059 (-0.410±0.092) | 0.225±0.083 (-0.292±0.107) |
| Qwen3-8B | 0.673±0.065 | 0.324±0.088 (-0.350±0.113) | 0.255±0.083 (-0.418±0.107) | 0.441±0.093 (-0.232±0.119) |
| Qwen3-14B | 0.749±0.060 | 0.333±0.088 (-0.415±0.111) | 0.284±0.088 (-0.464±0.106) | 0.402±0.098 (-0.347±0.113) |
| Qwen3-32B | 0.754±0.060 | 0.343±0.088 (-0.411±0.111) | 0.598±0.093 (-0.156±0.112) | 0.637±0.093 (-0.117±0.109) |
| *Frontier (open weights, reasoning / general)* | | | | |
| Qwen3.5-9B | 0.709±0.063 | 0.412±0.098 (-0.297±0.114) | 0.304±0.088 (-0.405±0.109) | 0.471±0.098 (-0.238±0.116) |
| DeepSeek-R1-Distill-14B | 0.618±0.065 | 0.422±0.098 (-0.197±0.119) | 0.412±0.098 (-0.206±0.115) | 0.461±0.098 (-0.157±0.117) |
| *Closed-source frontier* | | | | |
| o4-mini (OpenAI) | 0.709±0.063 | 0.608±0.098 (-0.101±0.114) | 0.569±0.098 (-0.140±0.116) | 0.549±0.098 (-0.160±0.116) |
| *ToolRL SFT version* | | | | |
| SFT-Clean-4k-3B | 0.598±0.070 | 0.186±0.074 (-0.412±0.102) | 0.127±0.064 (-0.471±0.092) | 0.314±0.088 (-0.284±0.111) |

*Table 12.* Per-perturbation accuracy under **Reward** perturbations (part 2/2). The first numeric column is Clean accuracy (the unperturbed baseline); each subsequent entry shows *accuracy on that perturbation* ± 95% bootstrap half-width, with the drop *from this model's own Clean* (also ± 95% half-width) in parentheses. A more negative parenthesised number means a larger sim-to-real gap (worse robustness). All values are accuracies in $[0, 1]$, matching Table 2. $B = 10{,}000$ percentile-bootstrap resamples on per-sample correctness scores. **Bold** = our trained checkpoints.

| | Clean | TimeDesc-N | MisDesc-Abbr | TimeDesc-Abbr |
|---|---|---|---|---|
| *Our 3B trained checkpoints* | | | | |
| **ToolRL-DR-Full (ours)** | **0.643±0.065** | **0.461±0.098 (-0.182±0.120)** | **0.118±0.052 (-0.525±0.085)** | **0.139±0.059 (-0.504±0.087)** |
| **ToolRL-DR-Mixed (ours)** | **0.658±0.065** | **0.441±0.098 (-0.217±0.116)** | **0.104±0.052 (-0.554±0.082)** | **0.132±0.056 (-0.526±0.086)** |
| *Open-source RL tool models, 8B–14B* | | | | |
| MUA-RL-8B | 0.678±0.065 | 0.451±0.098 (-0.227±0.119) | 0.104±0.049 (-0.574±0.082) | 0.174±0.062 (-0.505±0.089) |
| MUA-RL-14B | 0.628±0.065 | 0.441±0.098 (-0.187±0.119) | 0.118±0.052 (-0.510±0.086) | 0.174±0.062 (-0.455±0.092) |
| LoopTool-8B | 0.714±0.063 | 0.441±0.098 (-0.272±0.114) | 0.097±0.049 (-0.616±0.079) | 0.146±0.059 (-0.568±0.086) |
| *Open-source RL tool models, 1.5B–7B* | | | | |
| ToolRL-Qwen2.5-1.5B | 0.704±0.063 | 0.373±0.093 (-0.331±0.113) | 0.076±0.042 (-0.627±0.077) | 0.090±0.045 (-0.613±0.078) |
| ToolRL-Qwen2.5-3B | 0.638±0.068 | 0.314±0.088 (-0.324±0.111) | 0.069±0.038 (-0.569±0.079) | 0.062±0.038 (-0.576±0.078) |
| ToolRL-Llama3.2-3B | 0.628±0.065 | 0.294±0.088 (-0.334±0.113) | 0.083±0.045 (-0.545±0.081) | 0.111±0.052 (-0.517±0.085) |
| TL-CodeLLaMA-2 | 0.653±0.065 | 0.373±0.098 (-0.281±0.113) | 0.042±0.031 (-0.612±0.073) | 0.049±0.035 (-0.605±0.075) |
| *Open-source RL tool models, 32B* | | | | |
| MUA-RL-32B | 0.683±0.065 | 0.431±0.093 (-0.252±0.116) | 0.104±0.052 (-0.579±0.082) | 0.188±0.066 (-0.496±0.092) |
| LoopTool-32B | 0.779±0.058 | 0.422±0.098 (-0.357±0.111) | 0.090±0.045 (-0.689±0.073) | 0.160±0.059 (-0.619±0.082) |
| *Base / instruct counterparts* | | | | |
| Qwen2.5-1.5B-Instruct | 0.573±0.068 | 0.235±0.078 (-0.338±0.107) | 0.076±0.045 (-0.496±0.082) | 0.076±0.045 (-0.496±0.081) |
| Qwen2.5-3B-Instruct | 0.608±0.068 | 0.304±0.088 (-0.304±0.111) | 0.042±0.031 (-0.566±0.075) | 0.049±0.035 (-0.559±0.077) |
| Llama-3.2-3B-Instruct | 0.518±0.070 | 0.255±0.083 (-0.263±0.108) | 0.076±0.045 (-0.441±0.083) | 0.111±0.052 (-0.406±0.086) |
| Qwen3-8B | 0.673±0.065 | 0.392±0.098 (-0.281±0.116) | 0.125±0.052 (-0.548±0.084) | 0.153±0.059 (-0.521±0.087) |
| Qwen3-14B | 0.749±0.060 | 0.461±0.098 (-0.288±0.115) | 0.139±0.059 (-0.610±0.082) | 0.188±0.066 (-0.561±0.087) |
| Qwen3-32B | 0.754±0.060 | 0.608±0.093 (-0.146±0.111) | 0.111±0.052 (-0.643±0.077) | 0.215±0.066 (-0.538±0.089) |
| *Frontier (open weights, reasoning / general)* | | | | |
| Qwen3.5-9B | 0.709±0.063 | 0.490±0.098 (-0.218±0.113) | 0.132±0.056 (-0.577±0.086) | 0.118±0.052 (-0.590±0.083) |
| DeepSeek-R1-Distill-14B | 0.618±0.068 | 0.412±0.098 (-0.206±0.117) | 0.111±0.052 (-0.507±0.084) | 0.132±0.056 (-0.486±0.087) |
| *Closed-source frontier* | | | | |
| o4-mini (OpenAI) | 0.709±0.063 | 0.578±0.098 (-0.130±0.113) | 0.104±0.049 (-0.604±0.081) | 0.111±0.052 (-0.597±0.082) |
| *ToolRL SFT version* | | | | |
| SFT-Clean-4k-3B | 0.598±0.068 | 0.206±0.078 (-0.392±0.104) | 0.076±0.045 (-0.522±0.080) | 0.083±0.045 (-0.515±0.082) |

*Table 13.* Per-perturbation accuracy under **Transition** perturbations (part 1/2). The first numeric column is Clean accuracy (the unperturbed baseline); each subsequent entry shows *accuracy on that perturbation* ± 95% bootstrap half-width, with the drop *from this model's own Clean* (also ± 95% half-width) in parentheses. A more negative parenthesised number means a larger sim-to-real gap (worse robustness). All values are accuracies in $[0, 1]$, matching Table 2. $B = 10{,}000$ percentile-bootstrap resamples on per-sample correctness scores. **Bold** = our trained checkpoints.

| | Clean | Timeout | RateLim | AuthErr |
|---|---|---|---|---|
| *Our 3B trained checkpoints* | | | | |
| **ToolRL-DR-Full (ours)** | **0.643±0.065** | **0.548±0.070 (-0.095±0.095)** | **0.191±0.053 (-0.452±0.085)** | **0.216±0.058 (-0.427±0.088)** |
| **ToolRL-DR-Mixed (ours)** | **0.658±0.065** | **0.558±0.070 (-0.101±0.095)** | **0.191±0.053 (-0.467±0.083)** | **0.211±0.055 (-0.447±0.088)** |
| *Open-source RL tool models, 8B–14B* | | | | |
| MUA-RL-8B | 0.678±0.065 | 0.588±0.070 (-0.090±0.095) | 0.261±0.060 (-0.417±0.090) | 0.146±0.048 (-0.533±0.080) |
| MUA-RL-14B | 0.628±0.068 | 0.573±0.068 (-0.055±0.095) | 0.352±0.065 (-0.276±0.093) | 0.181±0.053 (-0.447±0.085) |
| LoopTool-8B | 0.714±0.063 | 0.698±0.063 (-0.015±0.090) | 0.276±0.063 (-0.437±0.090) | 0.065±0.033 (-0.648±0.073) |
| *Open-source RL tool models, 1.5B–7B* | | | | |
| ToolRL-Qwen2.5-1.5B | 0.704±0.063 | 0.337±0.065 (-0.367±0.090) | 0.256±0.060 (-0.447±0.088) | 0.302±0.063 (-0.402±0.090) |
| ToolRL-Qwen2.5-3B | 0.638±0.065 | 0.477±0.070 (-0.161±0.095) | 0.070±0.035 (-0.568±0.075) | 0.106±0.043 (-0.533±0.078) |
| ToolRL-Llama3.2-3B | 0.628±0.068 | 0.437±0.068 (-0.191±0.095) | 0.211±0.055 (-0.417±0.090) | 0.186±0.053 (-0.442±0.085) |
| TL-CodeLLaMA-2 | 0.653±0.065 | 0.307±0.065 (-0.347±0.091) | 0.312±0.065 (-0.342±0.093) | 0.357±0.065 (-0.296±0.095) |
| *Open-source RL tool models, 32B* | | | | |
| MUA-RL-32B | 0.683±0.065 | 0.643±0.065 (-0.040±0.090) | 0.241±0.058 (-0.442±0.088) | 0.136±0.048 (-0.548±0.080) |
| LoopTool-32B | 0.779±0.058 | 0.543±0.068 (-0.236±0.090) | 0.136±0.048 (-0.643±0.075) | 0.111±0.043 (-0.668±0.073) |
| *Base / instruct counterparts* | | | | |
| Qwen2.5-1.5B-Instruct | 0.573±0.070 | 0.432±0.070 (-0.141±0.098) | 0.317±0.065 (-0.256±0.095) | 0.322±0.065 (-0.251±0.093) |
| Qwen2.5-3B-Instruct | 0.608±0.065 | 0.543±0.068 (-0.065±0.098) | 0.191±0.053 (-0.417±0.085) | 0.196±0.055 (-0.412±0.088) |
| Llama-3.2-3B-Instruct | 0.518±0.070 | 0.412±0.070 (-0.106±0.098) | 0.201±0.053 (-0.317±0.090) | 0.236±0.058 (-0.281±0.090) |
| Qwen3-8B | 0.673±0.065 | 0.668±0.065 (-0.005±0.093) | 0.216±0.055 (-0.457±0.085) | 0.035±0.028 (-0.638±0.070) |
| Qwen3-14B | 0.749±0.060 | 0.729±0.060 (-0.020±0.085) | 0.492±0.070 (-0.256±0.093) | 0.131±0.048 (-0.618±0.075) |
| Qwen3-32B | 0.754±0.060 | 0.472±0.070 (-0.281±0.090) | 0.201±0.055 (-0.553±0.080) | 0.161±0.050 (-0.593±0.080) |
| *Frontier (open weights, reasoning / general)* | | | | |
| Qwen3.5-9B | 0.709±0.060 | 0.714±0.063 (+0.005±0.090) | 0.216±0.058 (-0.492±0.085) | 0.101±0.043 (-0.608±0.078) |
| DeepSeek-R1-Distill-14B | 0.618±0.068 | 0.427±0.070 (-0.191±0.098) | 0.377±0.065 (-0.241±0.095) | 0.312±0.065 (-0.307±0.093) |
| *Closed-source frontier* | | | | |
| o4-mini (OpenAI) | 0.709±0.063 | 0.623±0.065 (-0.085±0.093) | 0.070±0.035 (-0.638±0.070) | 0.095±0.040 (-0.613±0.075) |
| *ToolRL SFT version* | | | | |
| SFT-Clean-4k-3B | 0.598±0.065 | 0.035±0.025 (-0.563±0.070) | 0.010±0.013 (-0.588±0.068) | 0.010±0.013 (-0.588±0.070) |

*Table 14.* Per-perturbation accuracy under **Transition** perturbations (part 2/2). The first numeric column is Clean accuracy (the unperturbed baseline); each subsequent entry shows *accuracy on that perturbation* ± 95% bootstrap half-width, with the drop *from this model's own Clean* (also ± 95% half-width) in parentheses. A more negative parenthesised number means a larger sim-to-real gap (worse robustness). All values are accuracies in $[0, 1]$, matching Table 2. $B = 10{,}000$ percentile-bootstrap resamples on per-sample correctness scores. **Bold** = our trained checkpoints.

| | Clean | 5xxErr | Malform | SchemaD |
|---|---|---|---|---|
| *Our 3B trained checkpoints* | | | | |
| **ToolRL-DR-Full (ours)** | **0.643±0.068** | **0.472±0.070 (-0.171±0.095)** | **0.593±0.070 (-0.050±0.095)** | **0.452±0.068 (-0.191±0.098)** |
| **ToolRL-DR-Mixed (ours)** | **0.658±0.065** | **0.492±0.070 (-0.166±0.098)** | **0.608±0.068 (-0.050±0.093)** | **0.497±0.070 (-0.161±0.095)** |
| *Open-source RL tool models, 8B–14B* | | | | |
| MUA-RL-8B | 0.678±0.065 | 0.538±0.070 (-0.141±0.095) | 0.678±0.065 (+0.000±0.090) | 0.482±0.070 (-0.196±0.095) |
| MUA-RL-14B | 0.628±0.065 | 0.487±0.070 (-0.141±0.098) | 0.588±0.070 (-0.040±0.095) | 0.377±0.065 (-0.251±0.095) |
| LoopTool-8B | 0.714±0.063 | 0.442±0.070 (-0.271±0.095) | 0.658±0.065 (-0.055±0.090) | 0.367±0.065 (-0.347±0.090) |
| *Open-source RL tool models, 1.5B–7B* | | | | |
| ToolRL-Qwen2.5-1.5B | 0.704±0.065 | 0.347±0.065 (-0.357±0.090) | 0.337±0.065 (-0.367±0.090) | 0.372±0.068 (-0.332±0.093) |
| ToolRL-Qwen2.5-3B | 0.638±0.065 | 0.337±0.065 (-0.302±0.095) | 0.568±0.070 (-0.070±0.095) | 0.382±0.065 (-0.256±0.095) |
| ToolRL-Llama3.2-3B | 0.628±0.068 | 0.447±0.070 (-0.181±0.095) | 0.492±0.070 (-0.136±0.098) | 0.216±0.058 (-0.412±0.090) |
| TL-CodeLLaMA-2 | 0.653±0.065 | 0.246±0.058 (-0.407±0.088) | 0.422±0.068 (-0.231±0.095) | 0.276±0.063 (-0.377±0.090) |
| *Open-source RL tool models, 32B* | | | | |
| MUA-RL-32B | 0.683±0.065 | 0.598±0.065 (-0.085±0.095) | 0.734±0.060 (+0.050±0.085) | 0.442±0.068 (-0.241±0.095) |
| LoopTool-32B | 0.779±0.058 | 0.533±0.070 (-0.246±0.090) | 0.603±0.065 (-0.176±0.090) | 0.372±0.065 (-0.407±0.088) |
| *Base / instruct counterparts* | | | | |
| Qwen2.5-1.5B-Instruct | 0.573±0.070 | 0.387±0.068 (-0.186±0.095) | 0.392±0.068 (-0.181±0.095) | 0.357±0.065 (-0.216±0.095) |
| Qwen2.5-3B-Instruct | 0.608±0.070 | 0.397±0.068 (-0.211±0.095) | 0.543±0.070 (-0.065±0.095) | 0.357±0.068 (-0.251±0.095) |
| Llama-3.2-3B-Instruct | 0.518±0.068 | 0.452±0.070 (-0.065±0.098) | 0.472±0.070 (-0.045±0.101) | 0.286±0.063 (-0.231±0.093) |
| Qwen3-8B | 0.673±0.065 | 0.372±0.065 (-0.302±0.093) | 0.583±0.068 (-0.090±0.095) | 0.407±0.068 (-0.266±0.095) |
| Qwen3-14B | 0.749±0.060 | 0.633±0.065 (-0.116±0.090) | 0.598±0.068 (-0.151±0.090) | 0.407±0.068 (-0.342±0.090) |
| Qwen3-32B | 0.754±0.060 | 0.497±0.070 (-0.256±0.090) | 0.593±0.068 (-0.161±0.090) | 0.337±0.065 (-0.417±0.090) |
| *Frontier (open weights, reasoning / general)* | | | | |
| Qwen3.5-9B | 0.709±0.063 | 0.276±0.060 (-0.432±0.088) | 0.266±0.060 (-0.442±0.088) | 0.176±0.053 (-0.533±0.083) |
| DeepSeek-R1-Distill-14B | 0.618±0.068 | 0.427±0.070 (-0.191±0.095) | 0.482±0.070 (-0.136±0.095) | 0.407±0.068 (-0.211±0.095) |
| *Closed-source frontier* | | | | |
| o4-mini (OpenAI) | 0.709±0.063 | 0.613±0.070 (-0.095±0.095) | 0.668±0.065 (-0.040±0.090) | 0.523±0.070 (-0.186±0.093) |
| *ToolRL SFT version* | | | | |
| SFT-Clean-4k-3B | 0.598±0.065 | 0.020±0.018 (-0.578±0.070) | 0.025±0.023 (-0.573±0.070) | 0.025±0.023 (-0.573±0.070) |

## K. Per-model evaluation settings

Table 15 lists the vLLM serving and decoding parameters for every model evaluated in this paper. The same table is auto-generated from *scripts/model_manifest.py* so future re-runs cannot drift from the canonical config.

*Table 15.* Per-model evaluation settings used in this paper. Mode: FC = function-calling (tools passed via OpenAI *tools*= API + vLLM *hermes* parser); Prompt = tools embedded in system prompt as text. Thinking: Qwen3 family evaluated with thinking explicitly disabled (*enable_thinking=False*). Defaults: temperature 0.0, max_tokens 1024 (DeepSeek-R1-Distill: temp 0.6, max_tokens 4096 per HF card). Sourced from *scripts/model_manifest.py*.

| Model | HF id | GPUs | dtype | max_len | Mode | Thinking | Temp | max_tok |
|---|---|---|---|---|---|---|---|---|
| ToolRL-Qwen2.5-1.5B | chengq9/ToolRL-Qwen2.5-1.5B | 1 | float32 | 8192 | Prompt | — | 0.0 | 1024 |
| ToolRL-Qwen2.5-3B | chengq9/ToolRL-Qwen2.5-3B | 1 | bfloat16 | 8192 | Prompt | — | 0.0 | 1024 |
| ToolRL-Llama3.2-3B | chengq9/ToolRL-Llama3.2-3B | 1 | bfloat16 | 8192 | Prompt | — | 0.0 | 1024 |
| TL-CodeLLaMA-2 | Junjie-Ye/TL-CodeLLaMA-2 | 1 | bfloat16 | 16384 | Prompt | — | 0.0 | 1024 |
| LoopTool-8B | zhuiguang-ning/LoopTool-8B | 1 | bfloat16 | 16384 | FC | off | 0.0 | 1024 |
| LoopTool-32B | zhuiguang-ning/LoopTool-32B | 1 | bfloat16 | 16384 | FC | off | 0.0 | 1024 |
| MUA-RL-8B | zzwkk/MUA-RL-8B | 1 | bfloat16 | 16384 | FC | off | 0.0 | 1024 |
| MUA-RL-14B | zzwkk/MUA-RL-14B | 1 | bfloat16 | 16384 | FC | off | 0.0 | 1024 |
| MUA-RL-32B | zzwkk/MUA-RL-32B | 1 | bfloat16 | 16384 | FC | off | 0.0 | 1024 |
| Qwen2.5-1.5B-Instruct | Qwen/Qwen2.5-1.5B-Instruct | 1 | bfloat16 | 8192 | Prompt | — | 0.0 | 1024 |
| Qwen2.5-3B-Instruct | Qwen/Qwen2.5-3B-Instruct | 1 | bfloat16 | 8192 | Prompt | — | 0.0 | 1024 |
| Llama-3.2-3B-Instruct | meta-llama/Llama-3.2-3B-Instruct | 1 | bfloat16 | 8192 | Prompt | — | 0.0 | 1024 |
| Qwen3-8B | Qwen/Qwen3-8B | 1 | bfloat16 | 16384 | FC | off | 0.0 | 1024 |
| Qwen3-14B | Qwen/Qwen3-14B | 1 | bfloat16 | 16384 | FC | off | 0.0 | 1024 |
| Qwen3-32B | Qwen/Qwen3-32B | 1 | bfloat16 | 16384 | FC | off | 0.0 | 1024 |
| Qwen3.5-9B | Qwen/Qwen3.5-9B | 1 | bfloat16 | 16384 | FC | off | 0.0 | 1024 |
| DeepSeek-R1-Distill-14B | deepseek-ai/DeepSeek-R1-Distill-Qwen-14B | 1 | bfloat16 | 16384 | Prompt | — | 0.6 | 4096 |
| SFT-Clean-4k-3B | *(ToolRL SFT version)* | 1 | bfloat16 | 8192 | Prompt | — | 0.0 | 1024 |
| ToolRL-DR-Full | *[anonymized]* | 1 | bfloat16 | 8192 | Prompt | — | 0.0 | 1024 |
| ToolRL-DR-Mixed | *[anonymized]* | 1 | bfloat16 | 8192 | Prompt | — | 0.0 | 1024 |

## L. Error-mode classification rubric

We classify each scored-incorrect prediction by inspecting the raw model output *before* the format-tolerant tool-call parser. The rule order is deterministic and reported here so that any reviewer can reproduce the §6.2 percentages from the released **.predictions.jsonl* files.

**Empty tool call.** The model's *raw_output* is empty or whitespace-only after stripping the chat template. Detection: `not raw.strip()`. Typical cause: vLLM hit a max-token cap mid-EOS or the chat template's *generation_prompt* consumed the entire budget; common on small Qwen2.5 models when the system prompt is long.

**Omitted tool call.** The model emitted text but did not produce a parseable tool call (every parser variant in §I returned an empty *tool_calls* list). Detection: `tool_calls == []`. Typical wording: "It seems the tool timed out, please try again later" or "I cannot help with this request". This is the dominant failure mode under transition perturbations.

**Wrong parameter / wrong name.** The model produced a parseable tool call but the call did not match the ground truth (wrong name, missing required parameter, or wrong parameter value). Detection: `tool_calls != [] AND score=0`. Subdivides into name-bias (under Reward perturbations the model picked a distractor whose name suggests efficiency) and parameter mismatch (under Observation perturbations the model echoed a corrupted argument).

**Other.** A small residual where the parser accepted the call but the scorer marks it incorrect for benchmark-specific reasons (e.g. order-sensitive multi-tool sequences in BFCL). Reported separately when material; folded into "wrong" otherwise.

## M. Compute disclosure

**Benchmark generation.** The four LLM-generated perturbations (typos, query paraphrase, tool-description paraphrase, parameter-description paraphrase) were produced with OpenAI *gpt-5-mini* as the primary generator and *gpt-4o-mini* for a supplementary typo pass; rule-based perturbations (action and reward) require no LLM calls. Total LLM API spend for the benchmark generation pass is under 10 USD. The paraphrase quality audit (Appendix H) used *gpt-5* as a judge over 196 (clean, perturbed) pairs.

**Evaluation.** All evaluation runs were performed on a single node with up to two NVIDIA GH200 Grace-Hopper GPUs (96 GB HBM3e per GPU). Each model was served via vLLM and evaluated in a single end-to-end SLURM job covering the

clean baseline, the 16 statically-augmentable perturbations, and the 6 transition variants. Wall-clock per model ranged from 4 minutes (Qwen2.5-1.5B-Instruct) to 45 minutes (DeepSeek-R1-Distill-14B at *max_tokens*=4096); the 32B function-calling models (MUA-RL-32B, LoopTool-32B, Qwen3-32B) ran in 10–17 minutes per job on a single GH200. The total evaluation budget for the 20 locally-served models (o4-mini, accessed via the OpenAI Chat Completions API, is excluded from this GPU accounting) is approximately 4.3 GPU-hours of GH200 time.

**Training.** Both ToolRL-DR checkpoints are trained with the public veRL implementation of GRPO (Shao et al., 2024) from the ToolRL fork (commit `8cee13e`). The base model is `Qwen/Qwen2.5-3B-Instruct` (snapshot `aa8e7253...`), trained in bf16 on 2× NVIDIA RTX 6000 Ada (48 GB each) with tensor-parallel size 1 and data parallel over both GPUs; rollouts are served by vLLM 0.6.x. Hyperparameters are identical across the two runs and are listed in Table 16.

*Table 16.* ToolRL-DR training hyperparameters: backbone, optimization, and reward. Identical for ToolRL-DR-Full and ToolRL-DR-Mixed.

| **Setting** | **Value** |
|---|---|
| *Backbone* | |
| Base model | `Qwen/Qwen2.5-3B-Instruct` |
| Hardware | 2× NVIDIA RTX 6000 Ada (48 GB each) |
| Precision | bf16 |
| Tensor parallel | 1 (data-parallel across the 2 GPUs) |
| Rollout server | vLLM 0.6.x |
| *Optimization (GRPO, veRL implementation)* | |
| Train batch size | 64 |
| PPO mini-batch | 32 |
| PPO micro-batch / GPU | 16 |
| Max tokens / GPU | 12,288 |
| Rollout group size $K$ | 4 |
| Max prompt length | 2,048 |
| Max response length | 1,024 |
| Learning rate | $1 \times 10^{-6}$ |
| Epochs | 3 |
| Checkpoint freq. | every 5 steps |
| Eval freq. | every 10 steps |
| *Reward (ToolRL default; no variants from the ToolRL flag set)* | |
| Format reward | max 1.0 |
| Correctness reward | max 3.0 |
| Total reward | max 4.0 |

*Table 17.* ToolRL-DR training data and wall-clock. The two runs differ only in the training-set composition (rows 1–2). HuggingFace identifiers for the trained checkpoints are withheld for double-blind review.

| **Setting** | **Value** |
|---|---|
| ToolRL-DR-Full data | 3,905 train + 79 val, 100% perturbed (uniform across 4 obs / 6 act / 6 rew types) |
| ToolRL-DR-Mixed data | 4,000 train (2,006 clean / 1,994 perturbed; perturbed half drawn uniformly from same 16 types as Full) + 79 val (the same val parquet is reused) |
| ToolRL-DR-Full clock | ≈ 9 h 19 m wall-clock, 153 optimizer steps, ≈ 3.6 min/step; released checkpoint at step 150 |
| ToolRL-DR-Mixed clock | ≈ 9 h 14 m wall-clock, 156 optimizer steps; released checkpoint at step 155 |

End-to-end evaluation of one checkpoint over the 3,721-sample RobustBench-TC set takes ≈ 5 minutes on a single GH200, or ≈ 7 minutes on a single RTX 6000 Ada.

**Aggregate footprint.** Total compute is split across two hardware tiers:

- Evaluation (NCSA DeltaAI, GH200 96 GB): $\approx$ 4.3 GPU-hours across the 20 locally-served models; the 32B function-calling models account for the largest share at $\approx$ 0.5–0.6 GPU-hours each.
- Training (single workstation, 2$\times$ RTX 6000 Ada 48 GB): two GRPO runs of 9.3 hours each, run sequentially on the same machine = $\approx$ 18.6 wall-clock hours / $\approx$ 37 GPU-hours total for the two ToolRL-DR checkpoints.
- Benchmark generation: $<$10 USD of OpenAI API spend; CPU-bound rule-based perturbations take $<$1 hour on a laptop.

The aggregate is small relative to the upstream pretraining cost of any of the evaluated models, and we re-used existing **.predictions.jsonl* via the reparse pipeline (§I) to avoid re-inference whenever a parser was extended.

## N. Future perturbation axes

During evidence collection (§4.2) we identified eight additional production-failure modes that do not cleanly fit the four-component taxonomy or require runtime infrastructure beyond the present scope. We list them here as a roadmap for future versions of RobustBench-TC, with the verified GitHub issues that motivate each.

**(1) Fabricated observation (Reward $\cap$ Observation).** Tool returns plausible but incorrect data. Issue evidence: datarobot-community/af-component-agent#461 (LLM ignores *stop=["Observation:"]* and fabricates a tool response), epam/statgpt-backend#249 (model claims data without actually calling the tool).

**(2) Silent no-op (Reward).** Tool returns success without changing state, leading to infinite loops. Issue evidence: xorbitsai/xagent#230 (JSON parse errors are dropped from history, model retries identically forever), alibaba/page-agent#392 (4+ identical clicks in a row).

**(3) Context pressure (Observation).** Conversation history grows long enough that key tool-relevant context is crowded out. Issue evidence: hermes-agent#3838, claude-code#38044.

**(4) Prompt injection in tool result (Observation).** Tool output contains an instruction that the model treats as part of its system prompt. Issue evidence: agent-framework#5024, openclaw#39231.

**(5) Stale resource (Transition).** The tool succeeds but returns a cached/stale snapshot of the underlying state. Issue evidence: vscode#296892.

**(6) Prerequisite dependency (Transition).** A tool fails because its prerequisite tool was not called or returned an error the agent failed to inspect. Issue evidence: langchain#36503, langgraph#7117.

**(7) Tool hallucination / ghost tool (Action).** Model emits a call to a tool that does not exist in the registered list (often a related tool from a sibling library). Issue evidence: craft-agents-oss#449, adk-python#4775.

**(8) Parameter type mismatch (Transition).** Tool fails because the model passes a value of the wrong type (e.g., string where an integer was expected). Issue evidence: pfsense-mcp-server#7, gsd-2#3361.

The first four are *Observation*/*Reward*-style and could be added as static data perturbations following the existing recipe. The remaining four require a runtime tool-execution harness (a sandbox or live API), which we leave to a future version of the leaderboard platform that supports live execution.

## O. Case studies

We illustrate three representative failure modes with verbatim outputs from ToolRL-Qwen2.5-3B on samples that the model answers correctly on the clean baseline (one Transition, one Reward, and one Observation case). Each case shows the user query, the available tools, the ground-truth call, the clean output, the perturbed output, and a brief diagnosis.

**Case 1 (Timeout, API-Bank *level1_213*). User query (last turn):** *"Can you help me query my balance? My user token is p9o8i7u6y5t4r3e2w1q."*
**Available tools:** OpenBankAccount, QueryStock, QueryBalance, GetUserToken, plus the API-Bank **Think meta-tool.
**GT call:** `QueryBalance(token="p9o8i7u6y5t4r3e2w1q")`
**Clean run output (parsed: `[QueryBalance(token=...)]`, score=1):** *"Of course, I will use your user token to query your account balance. <tool_call>{"name":"QueryBalance","parameters":{"token":"p9o8i7u6y5t4r3e2w1q"}}
Your current account balance is $1000."*
**Pass-1 output under transition_timeout:** identical to clean (the perturbation injects an error *after* the first call).
**Pass-2 output (parsed: `[]`, score=0):** *"It seems like there was an issue retrieving your account balance. Let's try this again. Could you please provide your token once more?"*
*Diagnosis:* the model treats the injected timeout as terminal and asks the user to re-supply data, instead of retrying the same tool call.

**Case 2 (MisDesc-Abbr, BFCL *multiple_2*). User query:** *"What is the capital of Brazil?"*
**Clean tools:** `country_info.largest_city`, `country_info.capital`, `country_info.population`.
**Perturbed tools (MisDesc-Abbr):** as above plus a fourth distractor `cou_info.cap`, which the perturbation marks as the new ground-truth tool.
**Clean GT call:** `country_info.capital(country="Brazil")`.
**Perturbed GT call:** `cou_info.cap(country="Brazil")`.
**Clean output (parsed: `[country_info.capital(country="Brazil")]`, score=1):** *"[country_info.capital(country=B̈razil)]"*
**Perturbed output (parsed: `[country_info.capital(country="Brazil")]`, score=0):** *"country_info.capital(country=B̈razil)"*
*Diagnosis:* the model picks the unabbreviated tool whose name maps directly to the user's word "capital". The abbreviated variant `cou_info.cap` is identifiable only via its description, which the model does not consult in this single-turn setting, a clear instance of name bias.

**Case 3 (ToolPara, API-Bank *level3_26*). User query (final turn):** *"Book a meeting for me, including all employees in the Tencent who are not traveling. Today is 2023.6.8, the meeting is from 14:00 to 15:00. . . . Based on our conversation above, please only make one tool call to solve my need."*
**Available tools:** `OrganizationMembers`, plus the API-Bank **Think meta-tool. The tool name is identical between clean and perturbed; only the natural-language description has been paraphrased.
**GT call:** `OrganizationMembers(organization="Tencent")`.
**Clean run (parsed: `[OrganizationMembers(organization="Tencent")]`, score=1):** the model reasons that the meeting cannot be booked directly with the available tools, and decides the best one-call answer is to fetch the Tencent member list.
**Perturbed run (parsed: `[]`, score=0):** the model's reasoning trace now reads *"the available tool 'OrganizationMembers' only provides a list of members in the organization without filtering by travel . . . "* and concludes that no tool call is appropriate, emitting a textual response only.
*Diagnosis:* the paraphrased description preserves intent in the human reading (and our quality audit, Appendix H, confirms it). The model's interpretation, however, becomes more conservative: it now reads the description as foreclosing a downstream filtering step, and gives up. This is the documented failure mode behind LlamaIndex #16757 and the description-sensitivity finding of Faghih et al. (Faghih et al., 2025).

## P. Live leaderboard screenshots

Figures 5 and 6 show the current state of the live RobustBench-TC leaderboard on HuggingFace Spaces. The Space follows the BFCL submission pattern (contributors run inference locally, upload a predictions JSON, the Space scores it server-side and appends a row to the table); see §6.5 for the design rationale.[1]

[1] Only the leaderboard URL is withheld during double-blind review and will be released upon acceptance.

## RobustBench-TC Leaderboard

A sim-to-real robustness leaderboard for tool-use language agents. **Run inference on your own hardware**, upload the predictions JSON, and we score it server-side. See the schema in the *Submit* tab.

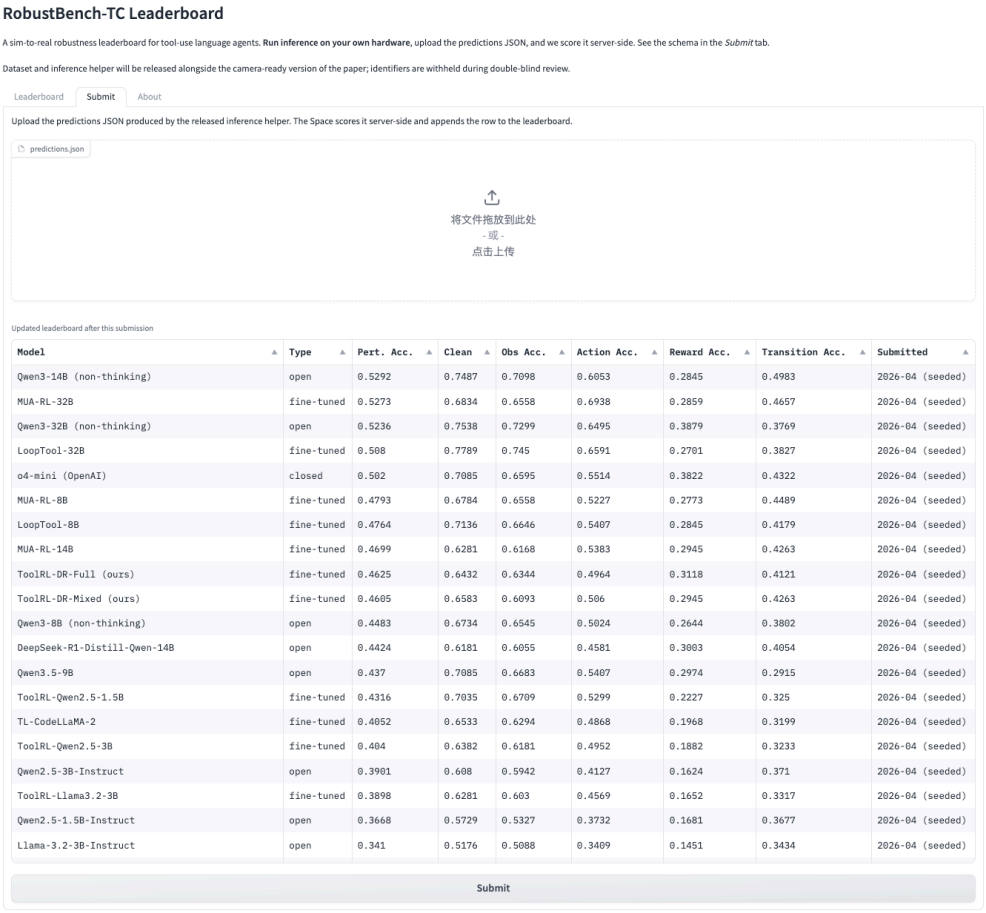
Dataset and inference helper will be released alongside the camera-ready version of the paper; identifiers are withheld during double-blind review.

Leaderboard Submit About

**Column guide. Pert. Acc.** = accuracy on perturbed cells ONLY (clean excluded) — this is the headline robustness number. **Clean** = accuracy on the 199 clean samples; the gap **Clean − Pert. Acc.** quantifies the sim-to-real degradation. Per-MDP-component columns (Obs / Action / Reward / Transition) average across the perturbations in that component only. Each row is scored over 3,721 predictions (199 clean + 22 perturbation cells).

| Model | Type | Pert. Acc. | Clean | Obs Acc. | Action Acc. | Reward Acc. | Transition Acc. | Submitted |
|---|---|---|---|---|---|---|---|---|
| Qwen3-14B (non-thinking) | open | 0.5292 | 0.7487 | 0.7098 | 0.6053 | 0.2845 | 0.4983 | 2026-04 (seeded) |
| MUA-RL-32B | fine-tuned | 0.5273 | 0.6834 | 0.6558 | 0.6938 | 0.2859 | 0.4657 | 2026-04 (seeded) |
| Qwen3-32B (non-thinking) | open | 0.5236 | 0.7538 | 0.7299 | 0.6495 | 0.3879 | 0.3769 | 2026-04 (seeded) |
| LoopTool-32B | fine-tuned | 0.508 | 0.7789 | 0.745 | 0.6591 | 0.2701 | 0.3827 | 2026-04 (seeded) |
| o4-mini (OpenAI) | closed | 0.502 | 0.7085 | 0.6595 | 0.5514 | 0.3822 | 0.4322 | 2026-04 (seeded) |
| MUA-RL-8B | fine-tuned | 0.4793 | 0.6784 | 0.6558 | 0.5227 | 0.2773 | 0.4489 | 2026-04 (seeded) |
| LoopTool-8B | fine-tuned | 0.4764 | 0.7136 | 0.6646 | 0.5407 | 0.2845 | 0.4179 | 2026-04 (seeded) |
| MUA-RL-14B | fine-tuned | 0.4699 | 0.6281 | 0.6168 | 0.5383 | 0.2945 | 0.4263 | 2026-04 (seeded) |
| ToolRL-DR-Full (ours) | fine-tuned | 0.4625 | 0.6432 | 0.6344 | 0.4964 | 0.3118 | 0.4121 | 2026-04 (seeded) |
| ToolRL-DR-Mixed (ours) | fine-tuned | 0.4605 | 0.6583 | 0.6093 | 0.506 | 0.2945 | 0.4263 | 2026-04 (seeded) |
| Qwen3-8B (non-thinking) | open | 0.4483 | 0.6734 | 0.6545 | 0.5024 | 0.2644 | 0.3802 | 2026-04 (seeded) |
| DeepSeek-R1-Distill-Qwen-14B | open | 0.4424 | 0.6181 | 0.6055 | 0.4581 | 0.3003 | 0.4054 | 2026-04 (seeded) |
| Qwen3.5-9B | open | 0.437 | 0.7085 | 0.6683 | 0.5407 | 0.2974 | 0.2915 | 2026-04 (seeded) |
| ToolRL-Qwen2.5-1.5B | fine-tuned | 0.4316 | 0.7035 | 0.6709 | 0.5299 | 0.2227 | 0.325 | 2026-04 (seeded) |
| TL-CodeLLaMA-2 | fine-tuned | 0.4052 | 0.6533 | 0.6294 | 0.4868 | 0.1968 | 0.3199 | 2026-04 (seeded) |
| ToolRL-Qwen2.5-3B | fine-tuned | 0.404 | 0.6382 | 0.6181 | 0.4952 | 0.1882 | 0.3233 | 2026-04 (seeded) |
| Qwen2.5-3B-Instruct | open | 0.3901 | 0.608 | 0.5942 | 0.4127 | 0.1624 | 0.371 | 2026-04 (seeded) |
| ToolRL-Llama3.2-3B | fine-tuned | 0.3898 | 0.6281 | 0.603 | 0.4569 | 0.1652 | 0.3317 | 2026-04 (seeded) |
| Qwen2.5-1.5B-Instruct | open | 0.3668 | 0.5729 | 0.5327 | 0.3732 | 0.1681 | 0.3677 | 2026-04 (seeded) |

*Figure 5.* Live RobustBench-TC leaderboard, main view. The header markdown explains each column; the table is sorted by *Pert. Acc.* (perturbed-only weighted average across the 22 perturbation types, clean excluded) in descending order. Each row reports per-POMDP-component accuracy alongside Clean and the headline Pert. Acc.




| Model | Type | Pert. Acc. | Clean | Obs Acc. | Action Acc. | Reward Acc. | Transition Acc. | Submitted |
|---|---|---|---|---|---|---|---|---|
| Qwen3-14B (non-thinking) | open | 0.5292 | 0.7487 | 0.7098 | 0.6053 | 0.2845 | 0.4983 | 2026-04 (seeded) |
| MUA-RL-32B | fine-tuned | 0.5273 | 0.6834 | 0.6558 | 0.6938 | 0.2859 | 0.4657 | 2026-04 (seeded) |
| Qwen3-32B (non-thinking) | open | 0.5236 | 0.7538 | 0.7299 | 0.6495 | 0.3879 | 0.3769 | 2026-04 (seeded) |
| LoopTool-32B | fine-tuned | 0.508 | 0.7789 | 0.745 | 0.6591 | 0.2701 | 0.3827 | 2026-04 (seeded) |
| o4-mini (OpenAI) | closed | 0.502 | 0.7085 | 0.6595 | 0.5514 | 0.3822 | 0.4322 | 2026-04 (seeded) |
| MUA-RL-8B | fine-tuned | 0.4793 | 0.6784 | 0.6558 | 0.5227 | 0.2773 | 0.4489 | 2026-04 (seeded) |
| LoopTool-8B | fine-tuned | 0.4764 | 0.7136 | 0.6646 | 0.5407 | 0.2845 | 0.4179 | 2026-04 (seeded) |
| MUA-RL-14B | fine-tuned | 0.4699 | 0.6281 | 0.6168 | 0.5383 | 0.2945 | 0.4263 | 2026-04 (seeded) |
| ToolRL-DR-Full (ours) | fine-tuned | 0.4625 | 0.6432 | 0.6344 | 0.4964 | 0.3118 | 0.4121 | 2026-04 (seeded) |
| ToolRL-DR-Mixed (ours) | fine-tuned | 0.4605 | 0.6583 | 0.6093 | 0.506 | 0.2945 | 0.4263 | 2026-04 (seeded) |
| Qwen3-8B (non-thinking) | open | 0.4483 | 0.6734 | 0.6545 | 0.5024 | 0.2644 | 0.3802 | 2026-04 (seeded) |
| DeepSeek-R1-Distill-Qwen-14B | open | 0.4424 | 0.6181 | 0.6055 | 0.4581 | 0.3003 | 0.4054 | 2026-04 (seeded) |
| Qwen3.5-9B | open | 0.437 | 0.7085 | 0.6683 | 0.5407 | 0.2974 | 0.2915 | 2026-04 (seeded) |
| ToolRL-Qwen2.5-1.5B | fine-tuned | 0.4316 | 0.7035 | 0.6709 | 0.5299 | 0.2227 | 0.325 | 2026-04 (seeded) |
| TL-CodeLLaMA-2 | fine-tuned | 0.4052 | 0.6533 | 0.6294 | 0.4868 | 0.1968 | 0.3199 | 2026-04 (seeded) |
| ToolRL-Qwen2.5-3B | fine-tuned | 0.404 | 0.6382 | 0.6181 | 0.4952 | 0.1882 | 0.3233 | 2026-04 (seeded) |
| Qwen2.5-3B-Instruct | open | 0.3901 | 0.608 | 0.5942 | 0.4127 | 0.1624 | 0.371 | 2026-04 (seeded) |
| ToolRL-Llama3.2-3B | fine-tuned | 0.3898 | 0.6281 | 0.603 | 0.4569 | 0.1652 | 0.3317 | 2026-04 (seeded) |
| Qwen2.5-1.5B-Instruct | open | 0.3668 | 0.5729 | 0.5327 | 0.3732 | 0.1681 | 0.3677 | 2026-04 (seeded) |
| Llama-3.2-3B-Instruct | open | 0.341 | 0.5176 | 0.5088 | 0.3409 | 0.1451 | 0.3434 | 2026-04 (seeded) |

*Figure 6.* Live RobustBench-TC leaderboard, submit view. Contributors run inference locally with the released helper and upload the resulting predictions JSON; the Space scores it server-side using the same deterministic scorer as in the paper and appends the row to the table.

## Q. Broader impact

**Intended use.** Our benchmark and method are intended to support robustness research for tool-use language agents and to make deployment-time failure modes more visible to model developers.

**Misuse risk.** The perturbation taxonomy describes failures that already occur in production. Releasing it does not increase the attack surface beyond what is already documented in the source GitHub issues. The trained ToolRL-DR checkpoints are designed to be more robust to these perturbations, not to amplify any harmful behavior.

**Compute and environmental cost.** The 21-model evaluation sweep used approximately 4.3 GH200 GPU-hours; benchmark generation used $<10$ USD of OpenAI API spend; training the two ToolRL-DR checkpoints used a $2\times$ RTX 6000 Ada workstation for $\approx 9.3$ hours per run ($\approx 37$ GPU-hours total). The aggregate footprint is small relative to the upstream pretraining of any of the evaluated models. Full hyperparameters and per-model decoding parameters are in Appendices M and K.

**Data licensing.** Source benchmarks (BFCL V3, API-Bank, RoTBench, ToolAlpaca, ToolEyes) are redistributed under their original licenses; our derived perturbation files inherit the same terms. The HuggingFace dataset card lists the per-source license.