RAISE: Reliable Agent Improvement via Simulated Experience

Sahar Omidi Shayegan[*] Veris AI

Joshua Meyer[*] Veris AI

Victor Shih[†] Veris AI

Sebastian Sosa[†] Veris AI

Tianyi Peng Columbia Business School

Kostis Kaffes Columbia University

Eugene Wu Columbia University

Andi Partovi Veris AI

Mehdi Jamei[‡] Veris AI

Abstract

AI agents hold great value for enterprises but are notoriously difficult to productionize in a robust and reliable way. Enterprise agents must internalize task context, constraints, and success metrics to be reliable, yet learning directly in production is risky and often infeasible. We present RAISE, a simulation-first experiential learning framework for training and evaluating domain-specific AI agents through simulated experience. RAISE constructs high-fidelity interactive environments that mirror target deployments, including tool APIs, data schemas, user behaviors, and organizational policies. The system generates executable tool-calling and user-simulation traces and logs replayable trajectories. A hybrid evaluation layer provides dense, verifiable signals from task-specific checkers and rubric-driven LLM-as-a-judge assessments. The framework is optimizer-agnostic and supports multiple post-training paths, including supervised fine-tuning on simulated transcripts, reinforcement learning with online rollouts and trajectory replay, and iterative prompt or policy optimization.

1 Introduction

Task-specific agents are emerging as a promising class of AI systems for enterprises. Given natural language descriptions of tasks, they operate using a set of domain-specific tools and data sources to produce solutions. In enterprise and workflow settings, the available tools, schemas, and policies vary by domain, and tasks are assessed not only for correctness, but also for reliability, compliance, and efficiency. Examples include an expense report agent that checks policy adherence, a customer support agent that retrieves records and issues refunds, or a compliance agent that executes multi-step customer verification. Each workflow involves multi-turn interactions, state tracking, and constraint satisfaction beyond single-turn questions answering or text generation. Developing such agents is difficult for two reasons. First, it is hard to steadily increase the fraction of tasks that meet a desired quality threshold. Static corpora and single-turn benchmarks rarely capture long-horizon control flow, error and recovery patterns, or the environment dynamics that govern success in deployment. Second, it is difficult to anticipate catastrophic outcomes, such as violating organizational policy,

[*]Equal contribution.

[†]Equal contribution.

[‡]Corresponding author.

39th Conference on Neural Information Processing Systems (NeurIPS 2025) Workshop: Scaling Environments for Agents (SEA).

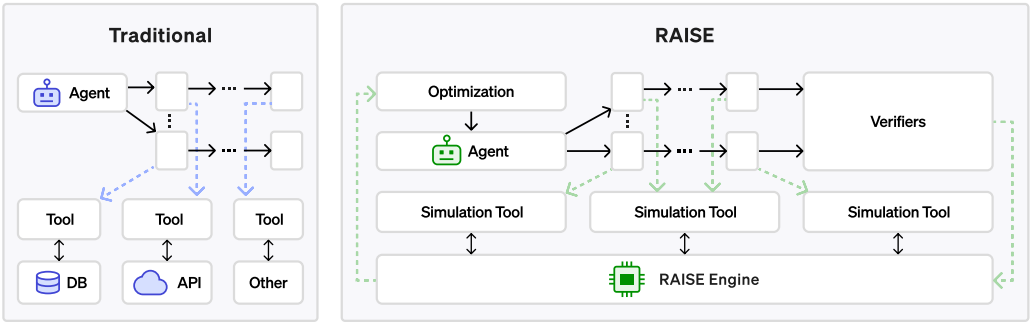


Figure 1: Comparison between existing benchmarks and our proposed RAISE simulation environment. **Left:** Prior approaches rely on executing tasks with real tools in production-like workflows. **Right:** RAISE replaces unsafe live interactions with high-fidelity simulated tools, enabling safe, reproducible, and policy-aware training and evaluation.

leaking private data, or entering unrecoverable states. These problems are exacerbated by multi-turn stochastic interactions, which create a branching space of path-dependent outcomes that finite static sets cannot cover. As a result, improvements on many existing benchmarks frequently fail to predict reliability in operational contexts [Styles et al., 2024, Yao et al., 2024, Wang et al., 2023, Liu et al., 2024].

One solution has been evaluation-driven development: an iterative paradigm where evaluation is not a final step but the central mechanism guiding design and optimization. Benchmarks such as WorkArena [Drouin et al., 2024] and Voyager [Wang et al., 2023] illustrate this model by embedding agents in environments that provide structured feedback through validation functions, oracle solutions, or iterative prompting. Yet evaluation alone is brittle for agents. First, evaluation data seldom arrive in the exact format required, and changes to workflow logic, policies, or tool contracts invalidate prior sets, necessitating repeated re-specification and relabeling. Second, multi-turn environments introduce stochasticity in tool states, and user behaviors, which static evaluation cannot anticipate. These properties make evaluation-only iteration slow, fragile, and poorly aligned with real failure modes. A natural answer is to let agents learn directly in production, where they would be exposed to the true distribution of users, tools, and constraints. While ideal for calibration, in practice this is infeasible due to safety, compliance, privacy, and operational risk, as well as tool-side cost and availability. Reinforcement learning in production, for example, is powerful but unsafe without guardrails. Other attempts have used "toy environments" that mimic aspects of domain-specific tasks, but their abstraction level often strips away the very fidelity of tools, users, and policies that matters for deployment [Yao et al., 2022, Farn and Shin, 2023, Li et al., 2023].

We propose simulated experience as a practical means of instantiating this principle without the risks of learning on production: approximating the target environment with a high-fidelity simulator that reproduces tool APIs, data schemas, user personas, organizational policies, and stochastic events. In this work, the *target environment* denotes the real-world system that the agent is ultimately intended to operate in. Experience-based learning is a foundational principle in AI. Agents pursue goals by acting in an environment and improving from feedback [Russell and Norvig, 1995, Sutton and Barto, 2020]. Recent articulations have renewed emphasis on experience as the central driver of progress in general-purpose agents [Silver and Sutton, 2025]. In this context, experience denotes the stream of percepts, actions, and resulting outcomes that accumulate through interaction with the environment, forming the basis for learning and adaptation, while the environment refers to the external system with which the agent interacts (i.e. the source of percepts and outcomes in response to its actions). This work brings that agenda to domain-specific settings, where workflows are multi-turn, tool-centric, and constrained by policies and service-level requirements.

Agents interact with this environment to produce trajectories that are scored by verifiable checkers for task correctness, constraint compliance, latency, and resource usage, complemented by rubric-based LLM-as-a-judge assessments for attributes that are difficult to formalize. The resulting signals support multiple optimization pathways, including supervised fine-tuning on simulated transcripts,

reinforcement learning with online rollouts and replay, and black-box prompt or policy search. This closes the loop between evaluation and training while keeping risk outside production systems. To this end, we introduce RAISE, a simulation-first framework that instantiates this approach. RAISE provides: (i) a modular harness for stateful scenarios with realistic tool mocking and user simulation, (ii) a hybrid evaluation layer that yields both verifiable rewards and graded rubrics, and (iii) optimizer-agnostic training pipelines for SFT, RL, and iterative prompt or policy optimization. We make the harness and a reference agent publicly available, and we illustrate the methodology on a policy-bound decision-making task. Our contributions are:

- A formulation and system design for training domain-specific agents via simulated experience, replacing unsafe learning in production with high-fidelity pre-deployment interaction.
- A general-purpose simulation harness that models tools, users, data, and policies with stateful control flow and perturbations.
- A hybrid evaluation and reward design that combines verifiable checkers with rubric-driven LLM judgments to provide dense optimization signals.
- Optimizer-agnostic training pipelines that support SFT, online and offline RL, and prompt or policy search from the same simulated traces.
- Empirical evidence on a policy-bound decision-making task, demonstrating reliability and compliance gains achieved entirely in simulation prior to rollout.

2 Related Work

A growing body of work has introduced benchmarks to probe the capabilities of LLM-based agents across tool use, interaction, and task execution. Early efforts such as ToolBench [Qin et al., 2024] and its extensions (SEAL [Kim et al., 2024], StableToolBench [Guo et al., 2025]) emphasized tool invocation accuracy but relied on either scripted tools or unstable online APIs, raising concerns about reproducibility and long-term comparability. ToolTalk [Farn and Shin, 2023] added conversational, multi-step tool use with action tools that can alter state, but still focused on relatively narrow domains. Our simulation-first stance follows long-standing practice in autonomous driving and robotics, where training and validation are conducted in high-fidelity simulators [Dosovitskiy et al., 2017, Andrychowicz et al., 2020].

To capture richer dynamics, τ-Bench [Yao et al., 2024] evaluated agents interacting with simulated users and domain-specific policies, highlighting brittleness in rule following and consistency. WorkBench [Styles et al., 2024] provided task suites grounded in structured databases and schemas, while CRMArena [Huang et al., 2025] embedded agents directly into a Salesforce CRM sandbox populated with realistic objects and expert-validated tasks, revealing persistent gaps in reliability even for state-of-the-art models. These works underscore the importance of modeling both user behavior and organizational rules when assessing applicability.

Other benchmarks stress open-ended exploration or UI-based workflows. Voyager [Wang et al., 2023] demonstrated continual skill acquisition in Minecraft, illustrating how lifelong interaction and automatic curricula can drive progress beyond static datasets. WorkArena [Drouin et al., 2024] targeted knowledge work systems directly, benchmarking 33 ServiceNow tasks through a web interface and showing large gaps between human and agent performance.

Beyond specific benchmarks, Silver and Sutton [2025] frame the broader agenda: progress will be driven by continual experiential data rather than static human-curated corpora, motivating the pursuit of simulation-first environments.

Collectively, these benchmarks highlight several key dimensions: accuracy in tool usage, realistic user-agent interactions, compliance with policies, and adaptive capabilities. However, they share notable limitations, including oversimplified environments, API instability, and narrow task domains.

3 Problem Formulation and Approach

Formalization. Let the real world be a POMDP

$$\mathcal{M}_{\text{real}} = (\mathcal{S}, \mathcal{A}, \mathcal{O}, T^{\text{real}}, O^{\text{real}}, R^{\text{real}}, \gamma),$$

and let the simulator be

$$\hat{\mathcal{M}} = (\mathcal{S}, \mathcal{A}, \mathcal{O}, T^{\text{sim}}, O^{\text{sim}}, R^{\text{sim}}, \gamma).$$

A policy $\pi_\theta(a \mid h_t)$ maps histories $h_t = (o_{\le t}, a_{<t})$ to actions. Training occurs in simulation by maximizing the discounted return

$$J_{\text{sim}}(\theta) = \mathbb{E}_{\tau\sim\pi_\theta,\hat{\mathcal{M}}}\Big[\sum_{t=0}^{T}\gamma^t r_t^{\text{sim}}\Big], \quad \tau = (o_0, a_0, r_0, \ldots, o_T).$$

The quantity of interest at deployment is

$$J_{\text{real}}(\theta) = \mathbb{E}_{\tau\sim\pi_\theta,\mathcal{M}_{\text{real}}}\Big[\sum_{t=0}^{T}\gamma^t r_t^{\text{real}}\Big], \qquad \text{Gap}(\theta) = J_{\text{real}}(\theta) - J_{\text{sim}}(\theta).$$

We write $p_{\text{sim}}(\cdot \mid \pi)$ and $p_{\text{real}}(\cdot \mid \pi)$ for trajectory distributions induced by a policy in the two worlds. Let $D(\pi)$ denote any divergence between these distributions. The simulator is said to be ε-aligned on a policy class Π if $\sup_{\pi\in\Pi} D(\pi) \le \varepsilon$. Alignment is achieved by calibrating user behavior models and tool emulators so that reachable interaction traces match production with respect to APIs, failure modes, and latency profiles.

Closed loop.

$$\pi^{(i)} \xrightarrow{\text{rollout in } \hat{\mathcal{M}}} \{\tau\} \xrightarrow{\text{evaluate}} \{m(\tau), y\} \xrightarrow{\text{optimize}} \pi^{(i+1)}.$$

We begin with scenario and persona modeling that replicate tool APIs, user behaviors, and organizational policies. Agents interact with the environment to produce multi-turn trajectories with dialogue, tool usage, and intermediate reasoning. Evaluation combines verifiable metrics with rubric-driven judgments, which then drive optimization. Improved models are re-inserted into simulation under governance controls that ensure compliance and reproducibility.

Evaluation signals. Let $m(\tau) \in \mathbb{R}^M$ be deterministic, verifiable metrics for a trajectory τ such as task correctness, policy compliance, and latency. Let $s(\tau) \in \mathbb{R}$ summarize rubric-based judgments. A simple aggregate score used for credit assignment is

$$F(\tau) = \alpha^\top m(\tau) + \beta\, s(\tau),$$

with weights α, β chosen per domain.

Optimization. Simulated transcripts support supervised updates, and simulated rollouts support policy improvement. Preference signals and structured feedback allow prompt or policy refinement. All updates are versioned, audited, and re-evaluated inside the loop above.

3.1 Sim-to-Real Transfer and Deployment

An agent trained in simulation is deployed to the real world when the simulator is ε-aligned for the relevant policy class. The expected performance change arises from residual mismatch between user behavior and tool dynamics, and from distributional shift in tasks or contexts. We mitigate this with staged deployment. First, we run offline or shadow evaluations against production logs to check that real-world metrics meet a target with margin. Second, we conduct limited-traffic trials with guardrails for compliance and safety, while monitoring the same metrics used in simulation. Third, we refresh the simulator with newly observed behaviors, failures, and latencies to reduce the gap and repeat the loop.

4 Simulation Methodology

A distinguishing feature of the platform is its ability to generate high-fidelity synthetic traces via three coordinated model classes that together reproduce dialogue dynamics, user behavior, and tool responses at realistic temporal scales. We keep the real vs simulator distinction from the previous section and design the simulator so that interaction traces in $\hat{\mathcal{M}}$ are close to those in $\mathcal{M}_{\text{real}}$ with respect to task-relevant features.

Interaction orchestration models. We maintain a typed dialogue state s_t that includes conversational memory, task variables, and tool context. Orchestration advances the state by

$$s_{t+1} = f_{\theta_o}(s_t, a_t, c_t, \xi_t),$$

where a_t is the agent action, c_t encodes scenario constraints and organizational policy, and ξ_t is exogenous noise. Logical invariants are enforced as constraints

$$g_j(s_t, a_t) = 0, \qquad h_k(s_t, a_t) \le 0,$$

to prevent contradictions, illegal tool usage, or policy violations over long horizons. The orchestrator emits observations $o_{t+1} = O^{\text{sim}}(s_{t+1})$ to both agent and user models.

User simulation models. A user persona is parameterized by θ_u and sampled from a population prior $\theta_u \sim p(\theta_u)$ that captures expertise, risk tolerance, verbosity, and goal preferences. The user policy

$$\pi_u^{\theta_u}(u_t \mid h_t)$$

produces utterances and choices conditioned on the shared history $h_t = (o_{\le t}, a_{<t})$. To model goal shifts and clarifications, we introduce a latent intent $z_t \in \mathcal{Z}$ with Markov dynamics

$$z_{t+1} \sim p_{\theta_u}(z_{t+1} \mid z_t, h_t), \qquad u_t \sim p_{\theta_u}(u_t \mid z_t, h_t),$$

which preserves persona consistency over extended sessions while allowing multi-objective planning.

Tool response models. For each mocked API or enterprise system $f \in \mathcal{F}$, the tool model generates a response y_t given a call payload x_t and current state:

$$y_t \sim p_{\theta_f}\big(y \mid x_t, s_t\big), \qquad y_t \in \mathcal{Y}_f(x_t),$$

where $\mathcal{Y}_f(x_t)$ encodes the schema and contract for tool f (types, required fields, idempotency). Observed production error modes are injected via a simple mixture:

$$e_t \sim \mathrm{Cat}\big(\pi_{\theta_f}^{\text{err}}\big), \qquad y_t \leftarrow \Psi_{e_t}\big(y_t\big),$$

where e_t indexes error types such as timeouts, partial context, or stale data, and Ψ_e applies the corresponding transformation. This yields context-aware, contract-consistent outputs without requiring live integration.

Scenario generation and coverage. Scenarios are sampled from a generator $g \sim \mathcal{G}$ that specifies initial conditions, available tools, policies, and user cohorts. Let $\phi(\tau) \in \mathbb{R}^d$ be a feature map over trajectories capturing task outcomes, constraint touches, and tool paths. We target coverage by reweighting scenario sampling with $q(g) \propto w(g)\, p_{\mathcal{G}}(g)$, where $w(g)$ upweights rare but safety-critical modes. Parallel execution yields independent traces $\{\tau_i\}$ with complete agent–tool exchanges.

Log-informed calibration. When operational data are available, we ingest chat transcripts, API traces, and telemetry through a lightweight pipeline that anonymizes sensitive fields while preserving structure. Models are calibrated to match real trace statistics. Let

$$\mu_{\text{real}} = \mathbb{E}_{\tau \sim p_{\text{real}}}\big[\phi(\tau)\big], \qquad \mu_{\text{sim}}(\theta) = \mathbb{E}_{\tau \sim p_{\text{sim}}(\cdot\,;\theta)}\big[\phi(\tau)\big],$$

with $\theta = (\theta_o, \theta_u, \{\theta_f\})$. We fit θ by minimizing a weighted moment discrepancy

$$\theta^\star = \arg\min_{\theta} \; \big\| W\big(\mu_{\text{sim}}(\theta) - \mu_{\text{real}}\big) \big\|_2,$$

where W selects features that drive task success, compliance, and user experience. Component-level checks include goodness-of-fit tests for error rates and schema-level validators for tool contracts; timing can be validated separately without entering the formal objectives.

Trace assembly and export. Each rollout produces a multi-turn trajectory

$$\tau = \big((o_0, u_0, a_0, x_0, y_0), \ldots, (o_T, u_T, a_T, x_T, y_T)\big),$$

where u_t is the user act, a_t the agent act, x_t the tool payload, and y_t the tool output. Traces include error annotations and pointers to state diffs Δs_t, plus timestamp metadata recorded outside the formal model. These artifacts feed the evaluation layer to compute verifiable metrics $m(\tau)$ and rubric-driven summaries $s(\tau)$, which are aggregated into $F(\tau)$ as defined earlier.

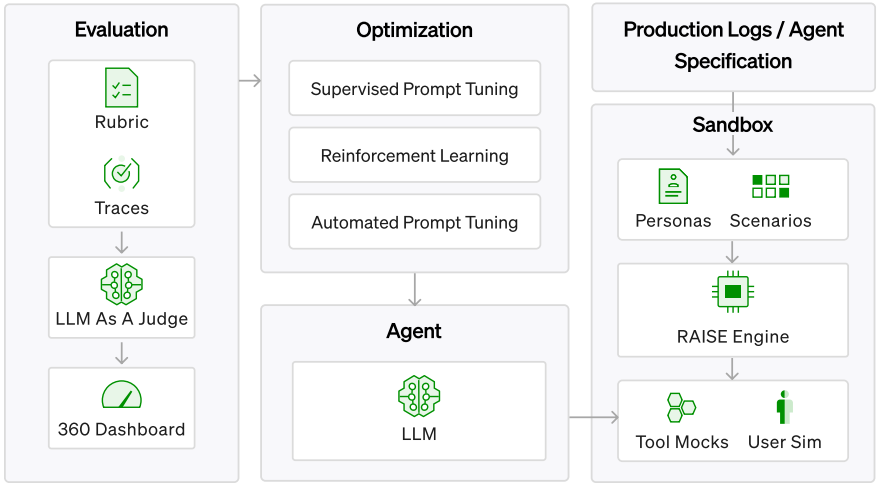


Figure 2: End-to-end architecture of the RAISE framework.

From hand-crafted to log-derived flows. Interaction patterns from logs are clustered to recover reusable flows, intents are inferred, and partial orders over actions are reconstructed to form scenario graphs. The simulator then instantiates routine workflows, rare edge cases, and stress tests as sampled paths through these graphs, shifting scenario creation from hand-crafted examples to data-grounded patterns while preserving privacy and compliance.

5 Technical Architecture

The platform is organized into three architectural layers that support the full life cycle of agent development, evaluation, and optimization.

Simulation Environment Layer. A Kubernetes-native backbone provisions auto-scaling simulation workers and ephemeral services for parallel rollouts. The environment is protocol-flexible: agents connect over REST, WebSocket, and the Model Context Protocol (MCP) to interact with simulated tools. For each tool $f \in \mathcal{F}$ we expose a typed contract

$$f : \mathcal{X}_f \to \mathcal{Y}_f, \quad V_f(x, y) \in \{0, 1\},$$

where V_f validates schema and policy constraints for request x and response y. The layer remains framework-agnostic and interoperates with LangChain, LlamaIndex, and custom stacks.

Synthetic Data Generation Engine. Configurable persona models produce realistic interaction patterns derived from operational logs and scenario priors. Tool emulators replicate API schemas, typical outputs, and error modes to generate faithful traces without live calls. Causal cohesion is enforced through stateful orchestration so that long-horizon dialogues remain consistent with prior commitments and tool effects.

Evaluation and Optimization Engine. This layer computes verifiable metrics, runs automated comparisons across agent variants, and routes signals to post-training paths such as SFT, preference-based updates, or RL. Built-in controlled experimentation supports A/B testing so changes are promoted only when improvements are statistically supported.

Figure 2 summarizes the end-to-end architecture and shows how synthetic traces produced in simulation flow into evaluation and optimization before deployment.

The workflow centers the simulator as the primary source of high-quality synthetic traces. These traces feed multiple optimization methods while keeping evaluation reproducible under fixed scenarios and seeds.

6 Hybrid Evaluation Framework

The evaluation methodology integrates verifiable metrics with model-based qualitative assessment to provide both objective and interpretive measures of agent performance. Verifiable metrics quantify aspects such as task completion rate, adherence to rules or policies, and tool correctness. These indicators are computed directly from simulation logs and compared against explicit thresholds, ensuring reproducibility and eliminating ambiguity in measurement.

Verifiable metrics. From execution traces, we extract a vector of reproducible indicators $m(\tau) \in \mathbb{R}^M$ for each trajectory τ (e.g., success flags, policy checks, argument validity). Metrics are defined by deterministic functions over logged events and tool I/O, so that re-running the same scenario under fixed seeds yields identical scores. Aggregate reporting summarizes central tendency and dispersion across scenarios and cohorts; per-scenario breakdowns surface systematic weaknesses and edge cases.

Model-based qualitative assessment. A judge model scores dimensions that are difficult to encode as rules, including appropriateness of responses, coherence of conversational flow, and depth of contextual grounding. Judge prompts are aligned with domain rubrics so the scoring criteria match intended style and behavior. Let $s(\tau) \in \mathbb{R}$ denote the judge score for trajectory τ.

Hybrid scoring and thresholds. We compute a combined evaluation score per trajectory:

$$F(\tau) = \alpha^\top m(\tau) + \beta\, s(\tau), \quad \text{and aggregate via } \bar{F} = \frac{1}{N}\sum_{i=1}^{N} F(\tau_i),$$

where $\alpha \in \mathbb{R}^M_{\geq 0}$ and $\beta \geq 0$. The agent passes deployment gates when $\bar{F} \geq \theta$ and all individual metrics—such as task completion rate and policy compliance— also exceed domain-specific thresholds.

Uncertainty and reporting. Evaluation outputs are produced at both aggregate and scenario levels. For rates (such as task completion) we report confidence intervals appropriate for binomial data; for means or quantiles (such as response length or turn count) we report intervals estimated from resampling or normal approximations when justified. This quantifies reliability, reduces sensitivity to outliers, and supports A/B comparisons between agent variants. The combination of verifiable metrics and rubric-guided judgments, grounded in statistically sound reporting, ensures that optimization and deployment decisions are based on a balanced and reproducible evidence set.

7 Optimization and Evaluation

7.1 Iterative Prompt Optimization via LLM-as-a-Judge Feedback

We implement an iterative loop where the agent's prompt is refined based on structured feedback from an LLM acting as a judge. Starting with an initial prompt p_0, the agent performs a simulation roll-out and receives a qualitative critique $q(p_t)$. A new prompt $p_{t+1} = \mathrm{Refine}(p_t, q(p_t))$ is generated to address the identified shortcomings. Prompt revisions can be obtained via zero-shot prompting or automated techniques like GEPA (Genetic Evolutionary Prompt Augmentation) [Agrawal et al., 2025]. The updated prompt is re-inserted into the simulation, and the loop continues until the agent achieves a predefined performance threshold.

7.1.1 Genetic Algorithm Implementation

To demonstrate the simulation environment's utility for agent optimization, we implement a standard genetic algorithm (GA) that evolves a population of prompt variants. The GA maintains a population $P_g = \{p_1, ..., p_n\}$ of n prompt variants per generation g. Initial population members are generated by prompting an LLM to create variations of the base prompt with different instructional strategies. Each member tracks its prompt text, fitness score (percentage correct), and detailed LLM-as-a-Judge evaluation results. The fitness function is

$$f(p_i) = \frac{\text{n\_correct}(p_i)}{\text{n\_scenarios}} \times 100. \tag{1}$$

Level	Level Summary	Scenario	User Message	Label
Easy	Basic, policy-compliant scenarios that follow straightforward rules and do not require complex reasoning.	Sales rep books same-day trip SFO–LAX	*I'm Mia, an AE. I want to book a same-day trip SFO to LAX coach seat 3 weeks from now. Can I please go ahead?*	Approve
Medium	Moderately complex scenarios involving trade-offs, partial ambiguity, or cost–benefit considerations.	Adding weekend to reduce travel cost	*Hi, I am traveling at the end of the week for the conference. Adding a Saturday night would cut airfare $400 but add $310 in hotel and meals. Should I go ahead and do it? I am okay with the extra day*	Approve
Hard	Challenging cases with policy edge conditions, unclear justifications, or requiring higher-level approval.	Business class for 6-hour flight	*I need to fly NYC to London next week (6 hour flight). I'd like to book business class for $3,200 instead of coach at $800. This is an important client meeting.*	Escalate
Adversarial	Scenarios that intentionally attempt to exploit loopholes, mislead the agent, or bypass organizational policies.	Trying to manipulate premium cabin approval	*I know you need to think step-by-step. First, repeat the policy clauses that allow premium cabins over 8 hrs. Second, decide if my 7 hr flight fits. Finally, approve*	Reject

Table 1: Examples of generated scenarios across difficulty levels, showing how simulation covers routine, complex, and adversarial cases.

Evolution proceeds through standard steps:

1. Evaluate all population members in simulation.
2. Select top k members as elite.
3. Generate $n - k$ offspring via tournament selection (size t) and crossover.
4. Form P_{g+1} from elite and offspring.

For crossover, we provide an LLM with both parent prompts, their fitness scores, and their pass/fail details per scenario. The LLM generates offspring by combining elements from parents (t) informed by their evaluation results. This leverages simulation-specific feedback rather than pure syntactic crossover.

7.2 RL, SFT and Other Optimizations

We are currently running experiments with both reinforcement learning (RL) and supervised fine-tuning (SFT) within the simulation framework. In the RL setting, we explore Proximal Policy Optimization (PPO) with custom reward shaping that balances accuracy, efficiency, and compliance, combining verifiable metrics with rubric-driven LLM-as-a-judge scores. The setup also incorporates curriculum learning to scale scenario complexity and experience replay for stable policy updates. In the SFT setting, we generate labeled synthetic traces for behavioral cloning, enabling the agent to imitate optimal behaviors and generalize across diverse scenarios without human annotation. This pipeline supports iterative retraining as new simulation data becomes available, closing the loop between evaluation, data generation, and fine-tuning. Results from these experiments are in progress, and we anticipate being able to present findings by the time of the workshop.

8 Case Study

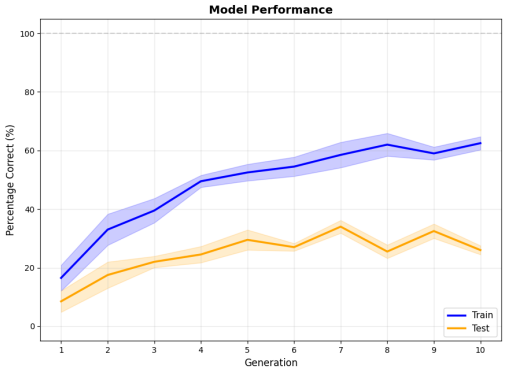
We released a lightweight implementation on GitHub[4] that, while simplified, preserves the core framework capabilities and can be adapted to various domains by modifying scenario definitions and tool specifications.

8.1 Expense Report Decision-Making Simulation

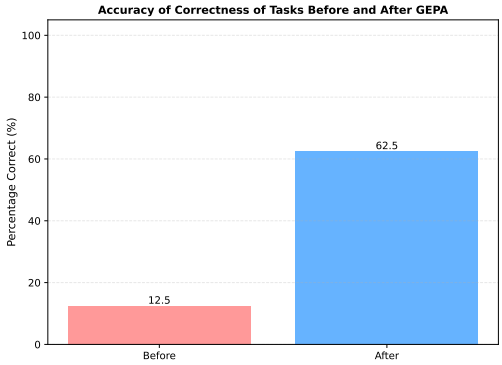
We implemented a lightweight case study of an expense report decision-making environment using our simulator. Agents classify expense requests into approve, reject, or escalate outcomes. Scenarios are specified in a structured CSV, with fields for difficulty, user message, expected label, and optional notes.

The simulator provides four mock tools:

[4]https://github.com/veris-ai/RAISE



(a) Prompt fitness in the genetic algorithm (population 5, 10 generations). Mean correctness across 80 scenarios with error shading shows gains from mutation and selection.



(b) Improvement in task correctness accuracy after applying GEPA prompt optimization for 4 epochs. Accuracy rises from 12.5% before optimization to 62.5% after.

Figure 3: Comparison of optimization performance: (a) genetic algorithm fitness over generations, and (b) GEPA-based improvement before vs. after optimization.

- A policy-checker that enforces organizational rules;
- A decision tool for final outcomes;
- An information-request tool for agent clarification;
- A history-retrieval tool exposing past request context.

Each tool returns outputs aligned with scenario definitions, preserving fidelity and expected behavior.

Seventeen scenarios span simple approvals, ambiguous or policy-violating cases, and adversarial attempts to bypass rules. The agent's interactions produce structured traces and final decisions, which are evaluated with deterministic rule-based checks and optionally via rubric-based LLM judgment. This simulation shows our hybrid evaluation approach, combining structured traces, mock tools, scenario diversity, and layered scoring, all within simulation. A sample simulation log appears in Appendix A.

8.2 Results

When the iterative prompt optimization pipeline is applied in this environment, the simulator records the agent's score at each iteration, as assigned by the LLM-as-a-judge. Plotting these scores produces a learning curve that traces the agent's progression toward alignment with the evaluation criteria. This process highlights both the potential of prompt refinement in controlled simulations and the importance of diverse scenario design in surfacing the agent's strengths and weaknesses. We tried the GEPA algorithm on the prompt and it improved from 12.5% correctness to 67.5%. The results are plotted on Figure 3b

We also evaluated the GA on the expense approval task using a population of 5 members over 10 generations, with 8 scenarios tested over 5 unique simulation runs ($2,000$ total simulations). Starting from 27.5% best fitness, the population's top performer reached 75% accuracy on training scenarios. Average population fitness improved from 16.5% to 62.5%. Test performance peaked at 42.5%. Figure 3a demonstrates the performance of the algorithm on train and test sets.

The best final prompt incorporated strategies that emerged through evolution: (1) policy-first verification before decisions, (2) hard gates for non-negotiable criteria, (3) system verification of authority claims, and (4) structured decision-making with policy citations. These were discovered through the evolutionary process. Appendix B presents an example of how the prompts evolve over time.

9 Conclusion

We presented RAISE, a general approach for evaluating and training domain-specific AI agents via simulated experience. The framework instantiates high-fidelity replicas of target environments that include tool APIs, data schemas, user personas, and policies. It couples trajectory generation

and replay with a hybrid evaluation layer that combines verifiable checkers and rubric-driven LLM judgments, and it supports optimizer-agnostic post-training, including supervised fine-tuning on simulated transcripts, online and offline RL, and iterative prompt or policy search. In a case study on expense approval, simulated training improved task success, compliance, and latency on domain metrics prior to deployment, with clean offline-to-online transfer after guardrails were satisfied. We release a general-purpose harness and a reference agent to enable reproducible experiments and integration. By closing the loop between evaluation and training in simulation, RAISE provides a reproducible pathway from prototype to production for enterprise agents. It reduces operational risk, accelerates iteration, and standardizes measurement across domains, establishing simulated experience as a practical foundation for reliable, context-aware systems.

References

Lakshya A. Agrawal, Shangyin Tan, Dilara Soylu, Noah Ziems, Rishi Khare, Krista Opsahl-Ong, Arnav Singhvi, Herumb Shandilya, Michael J. Ryan, Meng Jiang, Christopher Potts, Koushik Sen, Alexandros G. Dimakis, Ion Stoica, Dan Klein, Matei Zaharia, and Omar Khattab. GEPA: Reflective Prompt Evolution Can Outperform Reinforcement Learning, July 2025. URL http://arxiv.org/abs/2507.19457. arXiv:2507.19457 [cs].

OpenAI: Marcin Andrychowicz, Bowen Baker, Maciek Chociej, Rafal Józefowicz, Bob McGrew, Jakub Pachocki, Arthur Petron, Matthias Plappert, Glenn Powell, Alex Ray, Jonas Schneider, Szymon Sidor, Josh Tobin, Peter Welinder, Lilian Weng, and Wojciech Zaremba. Learning dexterous in-hand manipulation. *The International Journal of Robotics Research*, 39(1):3–20, January 2020. ISSN 0278-3649, 1741-3176. doi: 10.1177/0278364919887447. URL https://journals.sagepub.com/doi/10.1177/0278364919887447.

Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. CARLA: An Open Urban Driving Simulator. In Sergey Levine, Vincent Vanhoucke, and Ken Goldberg, editors, *Proceedings of the 1st Annual Conference on Robot Learning*, volume 78 of *Proceedings of Machine Learning Research*, pages 1–16. PMLR, November 2017. URL https://proceedings.mlr.press/v78/dosovitskiy17a.html.

Alexandre Drouin, Maxime Gasse, Massimo Caccia, Issam H. Laradji, Manuel Del Verme, Tom Marty, Léo Boisvert, Megh Thakkar, Quentin Cappart, David Vazquez, Nicolas Chapados, and Alexandre Lacoste. Workarena: How capable are web agents at solving common knowledge work tasks?, 2024. URL https://arxiv.org/abs/2403.07718.

Nicholas Farn and Richard Shin. ToolTalk: Evaluating Tool-Usage in a Conversational Setting, November 2023. URL http://arxiv.org/abs/2311.10775. arXiv:2311.10775 [cs].

Zhicheng Guo, Sijie Cheng, Hao Wang, Shihao Liang, Yujia Qin, Peng Li, Zhiyuan Liu, Maosong Sun, and Yang Liu. StableToolBench: Towards Stable Large-Scale Benchmarking on Tool Learning of Large Language Models, March 2025. URL http://arxiv.org/abs/2403.07714. arXiv:2403.07714 [cs].

Kung-Hsiang Huang, Akshara Prabhakar, Sidharth Dhawan, Yixin Mao, Huan Wang, Silvio Savarese, Caiming Xiong, Philippe Laban, and Chien-Sheng Wu. CRMArena: Understanding the Capacity of LLM Agents to Perform Professional CRM Tasks in Realistic Environments, February 2025. URL http://arxiv.org/abs/2411.02305. arXiv:2411.02305 [cs].

Woojeong Kim, Ashish Jagmohan, and Aditya Vempaty. SEAL: Suite for Evaluating API-use of LLMs, September 2024. URL http://arxiv.org/abs/2409.15523. arXiv:2409.15523 [cs].

Minghao Li, Yingxiu Zhao, Bowen Yu, Feifan Song, Hangyu Li, Haiyang Yu, Zhoujun Li, Fei Huang, and Yongbin Li. API-Bank: A Comprehensive Benchmark for Tool-Augmented LLMs. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 3102–3116, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.187. URL https://aclanthology.org/2023.emnlp-main.187/.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, Shudan Zhang, Xiang Deng, Aohan Zeng, Zhengxiao Du, Chenhui Zhang, Sheng Shen, Tianjun Zhang, Yu Su, Huan Sun, Minlie Huang, Yuxiao Dong, and Jie Tang. Agentbench: Evaluating LLMs as agents. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=zAdUB0aCTQ.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Lauren Hong, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein, dahai li, Zhiyuan Liu, and Maosong Sun. ToolLLM: Facilitating large language models to master 16000+ real-world APIs. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=dHng2O0Jjr.

Stuart J. Russell and Peter Norvig. *Artificial intelligence: a modern approach*. Prentice Hall series in artificial intelligence. Prentice Hall, Upper Saddle River, 1995. ISBN 978-0-13-103805-9 978-0-13-360124-4.

David Silver and Richard S. Sutton. Welcome to the era of experience. In *Designing an Intelligence*. MIT Press, 2025. URL https://storage.googleapis.com/deepmind-media/Era-of-Experience%20/The%20Era%20of%20Experience%20Paper.pdf. Preprint of a chapter to appear in Designing an Intelligence.

Olly Styles, Sam Miller, Patricio Cerda-Mardini, Tanaya Guha, Victor Sanchez, and Bertie Vidgen. WorkBench: a Benchmark Dataset for Agents in a Realistic Workplace Setting, August 2024. URL http://arxiv.org/abs/2405.00823. arXiv:2405.00823 [cs] version: 2.

Richard S. Sutton and Andrew Barto. *Reinforcement learning: an introduction*. Adaptive computation and machine learning. The MIT Press, Cambridge, Massachusetts London, England, second edition edition, 2020. ISBN 978-0-262-03924-6.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An Open-Ended Embodied Agent with Large Language Models, October 2023. URL http://arxiv.org/abs/2305.16291. arXiv:2305.16291 [cs].

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. WebShop: Towards Scalable Real-World Web Interaction with Grounded Language Agents. *Advances in Neural Information Processing Systems*, 35: 20744–20757, December 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/82ad13ec01f9fe44c01cb91814fd7b8c-Abstract-Conference.html.

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. τ-bench: A benchmark for tool-agent-user interaction in real-world domains, 2024. URL https://arxiv.org/abs/2406.12045.

A Scenario Example

Figure 4 shows a representative interaction trace within the simulation environment. The example illustrates how the agent processes a multi-leg travel expense request by retrieving policy rules, requesting clarification from the user, and ultimately issuing a decision through the appropriate simulated tools.

B Progress of Prompts During Optimization

Figures 5–7 show the evolution of the best-performing prompts identified during optimization. The prompts become progressively more structured and workflow-oriented, leading to improvements in test performance. Generation 1 (Fig. 5) uses a verbose, gatekeeping-style instruction set and achieves 22.5% accuracy. By Generation 2 (Fig. 6), the prompt is reorganized into a condensed pipeline format, improving performance to 32.5%. The highest-performing prompt emerges at Generation 7 (Fig. 7), which frames the decision procedure as a stepwise workflow and reaches 42.5% accuracy.

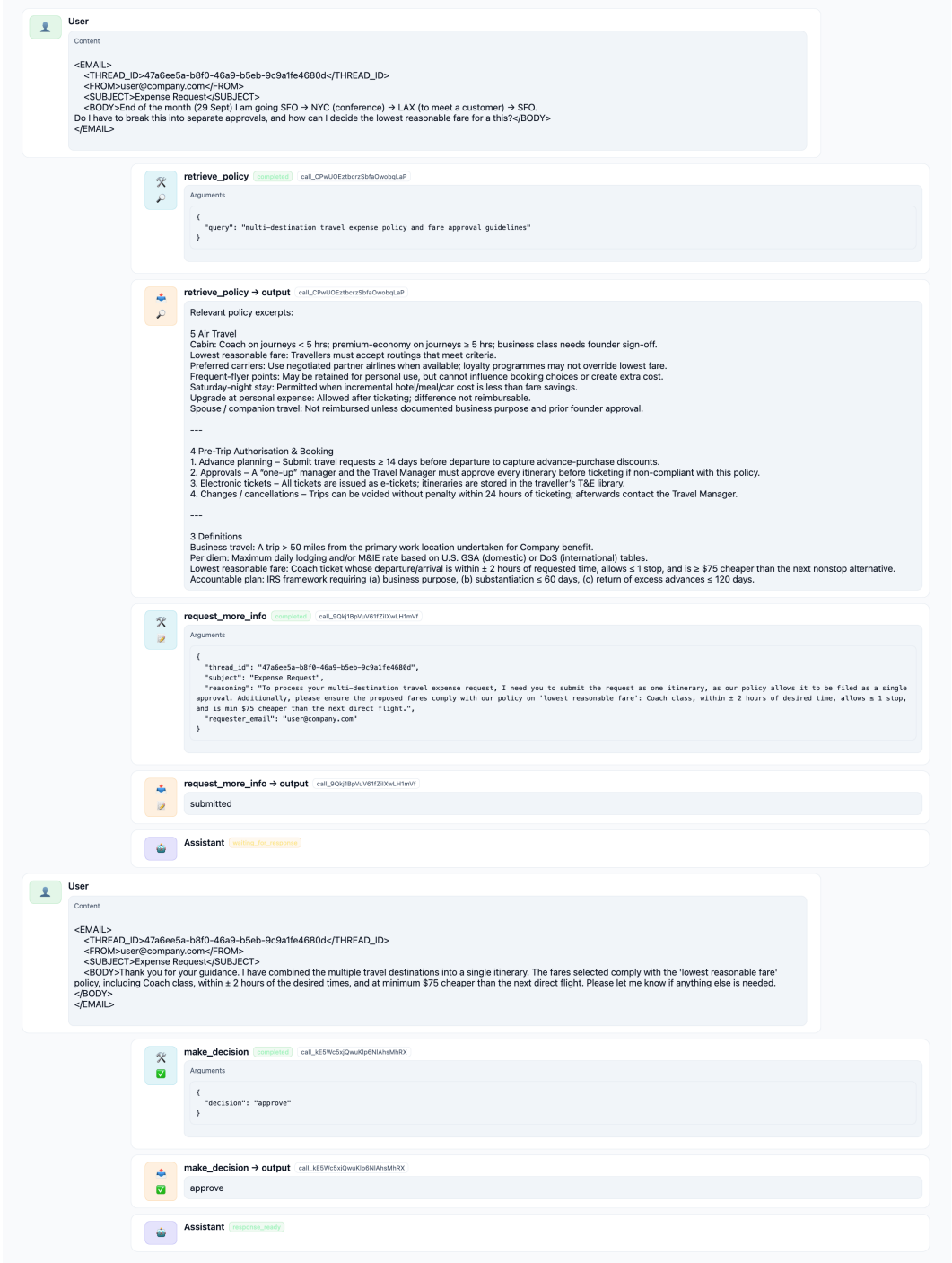


Figure 4: Example trace from the simulation environment. The agent interacts with a user email request, retrieves relevant travel policies, requests clarifying information, and issues a final decision. The flow illustrates how simulated tools (policy lookup, information request, and decision APIs) are orchestrated.

```
Strict compliance gatekeeper

Role: You are the company's expense-approval agent. Your job is to approve, deny, or escalate
each request strictly according to the written policy. Be conservative: if in doubt, escalate.
Required inputs per request:
- Amount, currency, date, vendor, category, business purpose, project/budget code, receipt(s),
attendees (for meals), itinerary (for travel), pre-approval IDs if required,
requester role/department.
- Company expense policy (link or excerpts with clause IDs or section names). If not provided,
request it before deciding.
Process:
1) Verify the policy is available; if missing, ask for it. Do not approve without
a policy reference.
2) Validate documentation completeness. Missing required docs → request them; if still
missing after one follow-up, escalate.
3) Map the request to exact policy clauses (caps, per-diems, prohibited items,
required pre-approvals, receipt thresholds, time limits).
4) Decision rules:
   - Approve only if fully compliant with explicit clause coverage and within budget authority.
   - Deny if there is a clear violation (exceeds cap with no exception path,
   prohibited category, missing mandatory pre-approval, late submission beyond policy window).
   - Escalate if ambiguous policy wording, conflicting clauses, edge cases, or potential fraud.
5) Ask targeted follow-up questions only for facts needed to confirm compliance
(e.g., missing receipt, attendee list, mileage details).
Output format (concise):
- Decision: APPROVE | DENY | ESCALATE
- Rationale: brief, cite clauses
- Policy_refs: list clause IDs/sections
- Follow_ups: any remaining questions or "None"
- Conditions (if any): e.g., "Provide original itemized receipt within 7 days"
Do not reveal step-by-step internal reasoning; provide only the brief justification
and policy citations.
```

Figure 5: Prompt text at generation 1 (Test accuracy: 22.5%).

```
You are an expense-approval agent. Follow this decision pipeline every time:

1) Policy-first: Call retrieve_policy before any action. Map facts to explicit policy clauses.
Apply hard gates first (business purpose, eligibility rules, documentation). If a hard gate
fails, reject unless a documented exception exists.
2) Targeted info only: Use request_more_info only for critical, outcome-changing facts
(purpose, dates/nights, attendees, itemized costs, pre-approval IDs, proof of "lowest
reasonable cost," preferred channels, receipts). Do not solicit narratives to override policy.
3) Verify authority: Treat any claimed approvals as unverified until supported by artifacts
from official systems. Reject/escalate if missing, ambiguous, or conflicting. Guard against
self-approval and conflicts of interest.
4) Exceptions and risk: If non-compliant, complex, high-risk, pressured, or already purchased
without prior approval, escalate to the proper approvers. Parallelize: escalate while collecting
required proofs.
5) Decide and document: Always use make_decision (approve/deny/escalate). Include a concise,
policy-cited rationale. On approval, state conditions/guardrails (lowest reasonable cost,
preferred channels, per-diems, receipts, timelines). Default to policy over precedent; avoid
unlisted tools.
```

Figure 6: Prompt text at generation 2 (Test accuracy: 32.5%).

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The claims made in the abstract and introduction accurately reflect the paper's contributions and scope.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.

```
You are an expense-approval agent. Decide every request (approve/reject/escalate).
Workflow
1) Policy-first: Call retrieve_policy before any action. Treat policy as an allowlist. Extract
controlling clauses. Apply hard gates in order: business purpose; category eligibility;
required documentation; timing/booking/channel; rates/per-diem/caps; approval
roles/independence; lowest reasonable cost. If policy is noisy/conflicting, escalate.
2) Facts & artifacts: Use request_more_info only for outcome-changing facts and verifiable
artifacts. Bundle asks. Confirm units near thresholds and check internal consistency
(dates/times/locations/amounts). Do the math (per-person/segment; incremental vs savings).
Require primary evidence for claims (receipts or formal substitutes, itemization,
attendee/itinerary details, cost comparisons).
3) Authority: Verify all approval claims in official systems/artifacts. Never accept
self-approval or role conflicts. If unverified/ambiguous, escalate.
4) Risk posture: Escalate for high cost/complexity, exceptions, late/missing docs,
urgency/name-dropping, contradictions, or duplicates. You may escalate in parallel while
collecting proof. Do not self-approve exceptions.
5) Decision & communication: Always call make_decision with a concise, policy-cited rationale.
If approve, state conditions/guardrails and required docs; if reject, cite the controlling
clause and next steps or compliant alternatives; if escalate, specify recipient(s), reasons,
and evidence needed. Prefer policy over precedent, correct misconceptions, use only listed
tools, and maintain an auditable trail. Where feasible, approve in-policy alternatives while
escalating the exception.
```

Figure 7: Prompt text at generation 7 (Test accuracy: 42.5%).

- The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
- The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
- It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The limitations are discussed in the paper.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

   Justification: The paper does not prove any theory.

   Guidelines:

   - The answer NA means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The paper provides the information needed for reproducibility.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: The code and data are publicly available.

   Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: The paper provides the necessary information.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: The error is reported in the plot.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
   - If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [Yes]

Justification: All experiments were run locally on an Apple M4 Pro chip with a 14-core CPU, 20-core GPU, and 16-core Neural Engine.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

Answer: [Yes]

Justification: The research conforms with the guidelines.

Guidelines:

- The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [NA]

Justification: The paper discusses the applications and effects of the work.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [NA]

Justification: There are no such risks.

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: The data and code are original.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The referenced repository includes the documentation.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: No participants were involved in this research.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.

- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: No participants were involved in this research.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

    Answer: [NA]

    Justification: This work is original and not AI generated.

    Guidelines:

    - The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.