TutorGym: A Testbed for Evaluating AI Agents as Tutors and Students

Daniel Weitekamp*[0000−0003−0079−8000],
Momin N. Siddiqui*[0000−0003−1874−7789], and
Christopher J. MacLellan[0000−0003−3084−5189]

Georgia Institute of Technology, Atlanta, GA 30332, USA
{weitekamp, msiddiqui66, cmaclell}@gatech.edu
(*) Equal contribution

Abstract. Recent improvements in large language model (LLM) performance on academic benchmarks, such as MATH and GSM8K, have emboldened their use as standalone tutors and as simulations of human learning. However, these new applications require more than evaluations of final solution generation. We introduce **TutorGym**[1] to evaluate these applications more directly. TutorGym is a standard interface for testing artificial intelligence (AI) agents within existing intelligent tutoring systems (ITS) that have been tested and refined in classroom studies, including Cognitive Tutors (CTAT), Apprentice Tutors, and OATutors. TutorGym is more than a simple problem-solution benchmark, it situates AI agents within the interactive interfaces of existing ITSs. At each step of problem-solving, AI agents are asked what they would do as a *tutor* or as a *learner*. As tutors, AI agents are prompted to provide tutoring support—such as generating examples, hints, and step-level correctness feedback—which can be evaluated directly against the adaptive step-by-step support provided by existing ITSs. As students, agents directly learn from ITS instruction, and their mistakes and learning trajectories can be compared to student data. TutorGym establishes a common framework for training and evaluating diverse AI agents, including LLMs, computational models of learning, and reinforcement learning agents, within a growing suite of learning environments. Currently, TutorGym includes 223 different tutor domains. In an initial evaluation, we find that current LLMs are poor at tutoring—none did better than chance at labeling incorrect actions, and next-step actions were correct only ∼52–70% of the time–but they could produce remarkably human-like learning curves when trained as students with in-context learning.

Keywords: Intelligent Tutoring Systems · AI Benchmarks · Simulated Learners · Large Language Models

1 Introduction

TutorGym is a system for interfacing several varieties of artificial intelligence (AI) agents—including large language models (LLMs), reinforcement learners

[1] **Website:** https://tutorgym.ai; **GitHub:** Teachable-AI-Lab/tutor_gym

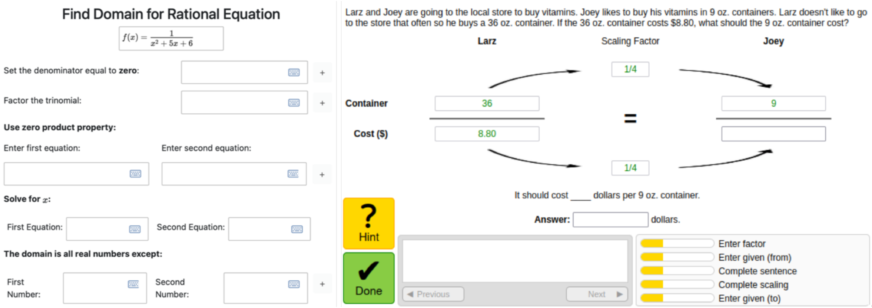
(RL), and computational models of learning—directly with existing intelligent tutoring systems (ITSs) including CTAT tutors [2], Apprentice Tutors [37], and OATutors [30]. TutorGym is inspired by Gym: an OpenAI system that enabled standardized comparisons between RL agents within several environments originally designed for human use, such as Atari games [5]. While Gym focuses largely on games and only on RL agents, TutorGym extends this concept to education, providing a standard method for evaluating several kinds of AI agents on tasks relevant to learning engineering and the learning sciences. It can evaluate the tutoring performance of AI agents and their ability to simulate student learning within ITSs that have been used in real classroom studies. TutorGym fills major gaps in prior benchmarks by aligning AI assessments with research-backed adaptive tutoring approaches and the data collected when applying those approaches in the classroom.

LLMs' ability to provide step-by-step solutions to academic tasks has ignited interest in using them as automated tutors [38]. No doubt, LLM capabilities for generating on-demand step-by-step solutions, explanations, and other content have compelling applications in automated tutoring. Nonetheless, blind enthusiasm for LLMs often understates the considerable new problems they introduce when applied to tutoring. LLM inference can be costly and consistently generates "hallucinations"—erroneous outputs that often sound plausible but are incorrect, logically inconsistent, or completely fabricated. Large-scale benchmarks of mathematical tasks like GSM8K [7] and the MATH dataset [11] measure LLM performance in generating final answers to math problems. High accuracies on these benchmarks (around 80-90%) mask poor real-world performance. For instance, the GSM-symbolic benchmark altered problems from GSM8K by replacing constants (e.g., names and numbers) and adding distractor clauses [25]. This substantially reduced LLM accuracy, suggesting that LLMs rely upon memorizing solutions instead of general-purpose problem-solving.

Instead of only evaluating correct final answer generation, TutorGym comprehensively evaluates AI agents on step-by-step problem-solving and step-by-step solution recognition within real ITSs interfaces. TutorGym's ITSs teach a variety of topics, and each permits a variety of solutions. Many of these have been meticulously refined through iterative deployment and redesign to provide precise adaptive tutoring that supports the practice of fine-grained skills.

1.1 Intelligent Tutoring Systems

ITSs are typically composed of at least three parts: 1) a tutor interface, 2) a tutor model, and 3) a student model [27]. Careful interface design is a core element of many historical ITS successes. These interfaces present students with *scaffolding*, where the cognitive problem-solving steps are distributed across several interface elements—boxes to fill, buttons to press, widgets to interact with—that each test particular well-defined skills as part of a step-by-step solution strategy (see Figure 1). This approach can be considerably more effective than simple non-adaptive assessments like online quizzes because the ITS can catch and address students' mistakes and lapses in knowledge at the step level



Find Domain for Rational Equation

$f(x) = \frac{1}{x^2 + 5x + 6}$

Set the denominator equal to **zero**:

Factor the trinomial:

Use zero product property:

Enter first equation:

Enter second equation:

Solve for x:

First Equation:

Second Equation:

The domain is all real numbers except:

First Number:

Second Number:

Larz and Joey are going to the local store to buy vitamins. Joey likes to buy his vitamins in 9 oz. containers. Larz doesn't like to go to the store that often so he buys a 36 oz. container. If the 36 oz. container costs $8.80, what should the 9 oz. container cost?

	Larz	Scaling Factor	Joey
		1/4	
Container	36	=	9
Cost ($)	8.80		
		1/4	

It should cost ____ dollars per 9 oz. container.

Answer: dollars.

Fig. 1. An Apprentice Tutor (left) and a CTAT tutor (right).

instead of only at the end of full problem-solving attempts [43]. The tutor model tracks students' step-by-step answers and aligns their problem-solving progress with a number of acceptable strategies. There are several varieties of tutor models. Constraint-based models characterize correct solutions by the satisfaction of several independent requirements, that when violated, trigger highly-specific feedback [26], conversational agents engage students in interactive dialogues [9] and approaches like example-tracing and model-tracing track students solutions step-by-step to deliver correctness feedback, hints, and other forms of support that are aligned with particular problem steps, solution strategies, and common mistakes [3]. In all of these cases, the role of the tutor model is to trace student problem-solving progress and adaptively align tutoring supports, such as on-demand hints, explanation messages, and step-by-step correctness feedback. Finally, the student model estimates students' mastery of individual knowledge components (facts, skills, or principles), to inform the selection of next practice problems and, in some cases, the behavior of the tutor model.

1.2 Evaluating AI Tutors with TutorGym

LLM-based tutoring is a tantalizing modern challenge in learning engineering that envisions LLMs as adaptive support tools instead of just—as is all too often the case—tools for cheating that bypass learning entirely. Prior work has addressed issues of LLM inaccuracy by limiting their use to offline content generation [6] or applying constraints to their on-demand generation [29]. Others have developed ways of using LLMs to translate static text resources into more constrained formats similar to those used by traditional tutoring paradigms [34] like AutoTutor chatbots [28]. Efforts like these rein in LLM inaccuracy. Yet far too few efforts provide a substantive comparison between LLM tutoring and the decades of research-based automated tutoring that have preceded it. In some cases, LLMs have been evaluated against human-generated hints [29], and group classroom instruction [13]. Yet, these efforts rarely justify how LLMs might innovate on the high standard set by prior work.

ITSs have been shown to be as effective as, and in some cases more effective than, human tutors [44,17]. Their main drawback is the labor intensity of authoring them, which can require as much as 200-300 hours of development per hour of instruction, and typically begins with investigations of expert knowledge and student misconceptions using a method called Cognitive Task Analysis (CTA) [2]. ITS designs are also often refined through analyses of student data [18]. These empirical elements of ITS development engage with the notion that there is no "magic bullet" to learning or tutoring [4,33]. Every capability requires mastery of several independent knowledge components [15], and effective tutoring requires precisely assessing and supporting mastery of these elements.

One may reasonably wonder if LLMs pose risks to learning engineers similar to those they pose to students. One cannot help but marvel when an LLM composes a step-by-step, textbook-like explanation before our eyes. However, on-demand assistance is not necessarily ideal or effective. Research-backed ITS design demands precision of understanding on the part of the learning engineer and precision of execution on the part of the ITS. It would be naïve to expect that the fruits of this understanding could be superseded by simply regenerating content from the non-adaptive datasets used to train LLMs. Yet perhaps the best of both is possible. Future AI research may deliver empirically backed cognitive supports, with the convenience of LLMs and the precision of traditional ITSs.

By interfacing various kinds of AI agents with a large repository of ITSs, TutorGym enables benchmarks of how precisely AI agents can deliver interface-based ITS instruction. Imagine a learning engineer who has done their homework: performed CTAs and designed a tutoring experience with a scaffolded interface. They may wish to skip the tedium of programming a tutor model and instead prompt an LLM to serve as a tutor model. TutorGym allows us to benchmark the feasibility of this scenario. Unlike the problem-solution pairs and sequential problem-solving that most LLMs are trained on, TutorGym environments deliver flexible adaptive tutoring. Many TutorGym tutors can support multiple solution strategies, allow for flexibility in step solution format and ordering, and utilize interface elements like buttons, checkboxes, and specialized widgets that deviate from purely text-based inputs. When students apply different sequences of actions in an ITS, they may end up in a wide variety of states that branch out from the initial problem presentation. For each of those states, each ITS in TutorGym can deliver step-specific hints and flexibly recognize a variety of possible correct next actions. In this work, we evaluate LLMs against tutor models in a large collection of intermediate tutor states. Specifically, in each state, we test if the AI can identify correct and incorrect next actions drawn from student data and demonstrate correct next actions.

1.3 Evaluating Simulated Learners with TutorGym

Artificial Intelligence in Education (AIED) researchers have long sought to simulate human learning for scientific and practical applications in education. John Self, the AIED Society founder, promoted the idea of "executable student models" that learn directly from tutors and operate as "wind tunnels" for comparing and investigating ITS designs [35,48]. Early simulated learner—like Sierra [40,41] and Cascade [42]—furthered these ideas, envisioning new applications where teachers practice tutoring with simulated learners [45], or students engage with them as learning companions [39,24]. More recent simulated learners—including SimStudent [16], Apprentice Learner [22,46], and AI2T [47]—can learn within tutor interfaces, inducing tutor models from minutes of interactive instruction. In line with Self's vision, several simulated learners have predicted experimental results [31,19] and reproduced student learning curves without fitting to student data [21,23,51].

A large body of work has replicated student behaviors and numerical changes in student performance without simulating learning directly [12,32]. LLMs have made it almost trivial to generate naturalistic human-like responses, meaning an undiscerning observer can be fooled by an LLM pretending to be a student. This has become a convenience in applications where the student "simulation" need only play the role of student, and not necessarily simulate learning.

Evaluating simulated learners—those that actually learn—involves comparing their evolving behavior to the responses of human students. By enabling AI agents to work directly within ITSs used in classroom studies, TutorGym expands upon the set of publicly available environments where this comparison can be made. A common way to evaluate simulated learners is to compare their learning curves to those of human learners. This is typically framed as a near-zero-parameter comparison: aside from small adjustments for student-specific individualization [23,51], models are not fit to student data, yet their *a priori* predictions must align with it [19,50]. These strict evaluations promote the view of simulated learners as theory-first computational models of learning [21] where the mechanisms of learning are theorized, implemented, and executed on learning materials, and only then compared to student data in a final evaluation [48]. Remarkably, computational models of learning have succeeded in closely aligning with human learning curves because they can learn from a few training experiences instead of the thousands to millions required by data-driven methods like reinforcement learning [19,20,49].

LLM-based simulated learners have not been directly compared with computational models of learning on learning curve reproduction; however, TutorGym provides the means of making such a comparison on a large scale. As an initial example of how this kind of comparison might look, we compare the learning curves of LLMs working in TutorGym environments with real student learning curves. In this evaluation, the LLMs "learn" using in-context learning [8]: their next-step answer predictions are graded as correct or incorrect, and these experiences accumulate within their context window.

2 The TutorGym API

TutorGym provides a standard Python interface for AI agents to interact with Cognitive Tutors (CTAT), Apprentice Tutors, and OATutors, and offers a common standard of interaction to unite these three distinct tutor platforms.

Table 1. Difference in functionality between tutor paradigms.

Tutoring Features	**CTAT Example-Tracing**	**Apprentice Tutors**	**OATutor**
Flexible Answers	yes	yes	yes
Unordered Actions	yes	partial	no
Optional Steps	yes	yes	partial
Tutor Performed Actions	yes	no	no
Contextual Control Flow	partial	partial	no
Accept Multiple Strategies	yes	no	no
Adaptive Scaffolding	partial	yes	partial
Adapt to Open-Ended Inputs	yes	yes	no
Tutor Any Problem Instance	partial	yes	partial
Knowledge Tracing	several	BKT	BKT

2.1 Tutor Paradigms

The Cognitive Tutor Authoring Tools (CTAT)[1], is the oldest of these tutoring platforms, and the most comprehensive in terms of domain diversity and flexible adaptive tutoring. CTAT interfaces include both standard HTML elements like text boxes, check boxes, and buttons; plus several specialized widgets like number lines and drag n' drop interfaces. CTAT has evolved through implementations in Java, Flash, and HTML. TutorGym supports CTAT's HTML-based tutors authored with example-tracing [2]: a programming-by-demonstration approach for authoring behavior graph-based tutor models. CTAT behavior graphs are essentially finite state machines with state nodes and action edges that can be augmented with additional functionality like unordered groups of actions, skippable actions, tutor-performed actions that trigger interface changes, and short programs that flexibly match variations in acceptable student answers.

Table 1 outlines how CTAT example-tracing, Apprentice Tutors, and OATutors compare on several dimensions of functionality. For instance, Apprentice tutors are implemented with hierarchical rules that divide tasks into scaffolded subtasks [36,37]. These rules are implemented with a Python-based rule engine that enables flexible control-flow and grading behavior. Symbolic algebra tools like SymPy can be used with these hierarchical rules to relax the formatting requirements of mathematical answers. In principle, rule-based tutoring models like Apprentice Tutors enable more complex behaviors than CTAT example-tracing. For instance, rule-based tutors can capture solution paths that vary considerably between problem instances, such as when tutors require students to engage in contextual decision-making and decide between different strategies. Nonetheless, the Apprentice Tutors included in TutorGym at the time of writing this paper consist of purely sequential steps. However, the granularity of those steps can be controlled dynamically as part of next-problem selection by selecting different "scaffold levels."

The within-problem control flow complexity of CTAT's tutors are considerably greater by comparison: they sometimes permit dozens of correct next actions, while the Apprentice tutors permit just one or two. Both of these paradigms include domains where open-ended student inputs affect the acceptable answers in later steps. All three tutor paradigms have tools for tutoring arbitrary problem instances. Apprentice Tutors can randomly generate problems and tutor them on demand, while OATutor and CTAT example-tracing provide tools for mass-producing problems by replacing placeholders in templates.

In terms of control flow complexity, OATutors are relatively simple by comparison to CTAT example-tracing and Apprentice Tutors. The OATutor problem pool consists mostly of one-step problems with some exceptions. However, short sequential "tutor pathways" offer a mixture of hints and extra scaffold steps that are available to students on demand [30]. OATutors permit flexible recognition of mathematical expressions, similar to Apprentice Tutors.

Adding new problems to TutorGym is as easy as copying standard tutor models from its three supported paradigms. At the time of writing, TutorGym includes 10 domains from CTAT's Mathtutor suite with 732 problems (and more on the way), 30 Apprentice Tutor domains with random problem generators, and 188 OATutor domains with 5161 problem instances. In addition, two miscellaneous hard-coded domains: fraction arithmetic and multi-column addition with random problem generators, are included from a limited early version of TutorGym [20]. These domains are implemented by programmatically generating finite-state machines using the same backend as our CTAT behavior graph interpreter. These tools are designed to be extensible to facilitate the integration of TutorGym with additional tutor paradigms in the future.

2.2 Tutor-AI Interfaces

Symbolic Interface TutorGym can adaptively tutor AI agents that implement two endpoints: `act(state)` and `train(state, action, reward)`. The `get_state()` endpoint exposes each tutor's current `state` as a `ProblemState` object containing a JSON-like representation of the tutor's current HTML interface. When a `state` is passed to the agent's `act(state)` endpoint, it returns the next `action` it will attempt in the tutor interface. Each `action` instance contains a `(selection, action_type, input)` triple (an SAI) that indicates 1) the `selection`: the `id` of

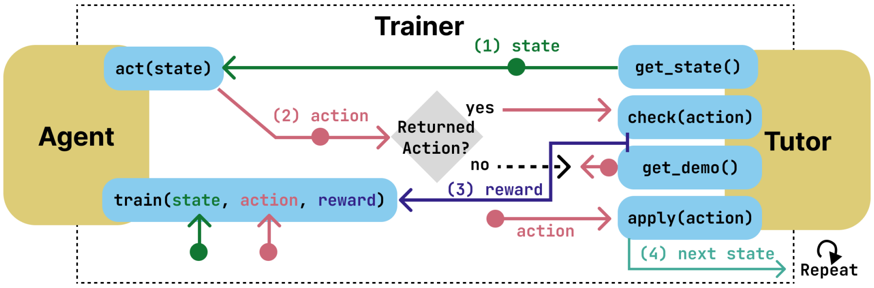


Fig. 2. The `Trainer` class training an AI agent with a tutor.

the interface element being acted on, 2) a string indicating what kind of action is being applied, and 3) the content of the action (usually a string). For instance, an SAI for an action to fill a text field with the value "7" might look like `("field1", "UpdateTextField", "7")`.

A `Trainer` (see Figure 2) class mediates the interactions between an agent and tutor, over a sequence of training problems. In each state the `Trainer` queries the agent's `act(state)` function and evaluates it with the tutor's `check(state, action)` function to return a numerical `reward` (1 for correct, -1 for incorrect), that is passed to `train(state, action, reward)` to train the agent. If the agent provides no next action because it does not know what to do next then the `Trainer` calls `get_demo(state)` in the tutor to instead demonstrate a 'bottom-out' hint to the agent, that is passed to `train(state, action, reward)` in the form of a worked example `action` with a `reward` of 1. Optionally, the `Trainer` can force a demo `action` after a specified number of incorrect actions—a helpful feature for forcing training to move on when stubborn agents repeatedly guess incorrectly. When a correct action is produced, the `apply(action)` function applies the action to the tutor, yielding a next state. Customizable logger classes can be provided to the `Trainer` to record each transaction. For instance, the `DataShopLogger` class records the standard transaction format used by PSLC DataShop [14].

The symbolic interface can be used directly by Python-based computational models of learning like the Apprentice Learner [22] and AI2T [47], and serves as the foundation for the LLM and reinforcement-learning interfaces.

LLM Interface The LLM interface is an agent wrapper that translates the `state`, `reward`, and next `action` demos from the symbolic interface into a prompt. By default the LLM is given the tutor state in JSON format. TutorGym can also open tutors in the browser and capture rendered images to augment the JSON-based state. The additional content of the LLM prompt is open-ended and up to researchers to design. In this work, we report an agent that implements `act(state)` by adding in-context examples collected from previous training.

Reinforcement Learning Interface TutorGym provides an `RL_Wrapper` class that can be applied to any tutor, and provides fixed state and action spaces that can be used as an OpenAI Gym environment for training reinforcement learners. This interface simply maps values in the state, and SAIs in actions to fixed integer values that can be used as action indicators and indices for one-hot state vectors that can be read by neural networks. Prior work [20,49] has shown that RL agents need tens of thousands of examples to succeed at tutoring tasks, and so in practice, only tutors that can randomly generate problem sets are compatible with the `RL_Wrapper`.

Tutoring Evaluation Support: Completeness Profiles Evaluating agents on open-ended tutoring requires a comprehensive assessment of behaviors beyond step-by-step problem-solving. For this purpose, TutorGym can automatically generate *completeness profiles*: collections of reachable ITS states with every correct next action produced by the tutor's `get_all_demos()` function. When student data is available, these profiles also include correct and incorrect student actions. Completeness profiles are built either by sampling several solution pathways in each problem of a large problem set or by replaying transactions from data to reproduce the tutor states experienced by students. An agent with 100% correct tutoring behavior should be able to accurately grade the completeness profile—labeling the correct actions as correct and the incorrect actions as incorrect—and generate at least one correct action as a demo for each state.

3 Tutoring Evaluation

3.1 Method

As an initial demonstration of using TutorGym to evaluate AI agents as interactive tutors, we tested several different LLMs, including Sonnet-3.5, Haiku-3.5, GPT-4o, and DeepSeek-v2.5 in TutorGym's 223 different tutor domains.[2] These commercial models offered the best performance vs. cost tradeoff at the time of writing. The chosen open-source models were the most performant models runnable with Ollama on our server with four A40 GPUs. For Apprentice Tutors we generated completeness profiles from a student dataset containing about 15 thousand transactions. These profiles include both the correct next actions from each tutor's `get_all_demos()` output and the correct and incorrect actions taken by students. We were unable to acquire public student data for the CTAT Mathtutors and OATutors. In these domains, the completeness profiles are generated by sampling tutoring paths from all of the available problems in each of the 10 CTAT tutors domains and 5 problems per tutor domain for the OATutors. We augmented all profiles with incorrect actions produced by DeepSeek-v2.5 (which we could run locally).

Recall that this evaluation simulates a scenario where a learning engineer prompts the LLM to operate as the ITS expert model in place of authoring a hard-coded one. To support this, we created specialized prompts that describe each tutor action with in-context examples and ask the agent to either generate a correct `action` for a given `state` (i.e., a worked example), or evaluate a given `state`–`action` pair by responding with "yes" if the action is correct or "no" if it is not.[3] OATutors and Apprentice tutors have well-defined steps that are fairly clear just from the labels and field ids in the problem state. For CTAT tutors, we supplement the prompt with domain-specific descriptions that explain their more complex control flows.[4]

3.2 Results

Table 2 shows the results of our evaluation for all tutors and LLMs. All four of the LLMs that we tested are quite poor or heavily biased when grading step-by-step action correctness. The models with the highest accuracy at identifying correct actions were the worst at identifying incorrect actions, and no model exceeded chance (i.e. 50%) at identifying incorrect actions. All models show poor

[2] The snapshots were: `claude-3-5-sonnet-20241022`, `claude-3-5-haiku-20241022`, `gpt-4o-2024-08-06`, and `deepseek-v2.5:236b`.

[3] see tutorgym/eval/prompts.yaml and tutorgym/eval/action_type_examples.yaml

[4] see tutorgym/eval/ctat_prompts.yaml

Table 2. LLM accuracy at labeling correct and incorrect student actions and generating next step demonstrations (bottom out hints) across the tutors.

Tutor Platform	**LLM Model**	**Correct Accuracy**	**Incorrect Accuracy**	**Demo Accuracy**
CTAT Mathtutors (10 domains)	Sonnet-3.5	61.92%	36.21%	**56.50%**
	Haiku-3.5	**81.06%**	25.05%	36.49%
	GPT-4o	28.11%	**42.63%**	38.89%
	DeepSeek-v2.5	59.96%	30.84%	39.33%
Apprentice (30 domains)	Sonnet-3.5	86.54%	46.56%	64.20%
	Haiku-3.5	**88.36%**	22.34%	49.73%
	GPT-4o	74.61%	**49.80%**	**70.75%**
	DeepSeek-v2.5	82.00%	48.92%	58.35%
OATutor (183 domains)	Sonnet-3.5	79.59%	**38.85%**	**52.10%**
	Haiku-3.5	**92.36%**	10.82%	36.63%
	GPT-4o	71.20%	38.69%	51.07%
	DeepSeek-v2.5	90.87%	17.21%	43.89%

accuracy—around 52-70%—at producing next-step demonstrations. These accuracies are starkly lower than many published final-answer benchmark results. Consistent with other work [10], they illustrate that current LLMs are impractical for providing out-of-the-box “complete” tutoring system support that grades and generates next actions in arbitrary intermediate tutor states.

3.3 Discussion

The relatively poor performance of LLMs at these tutoring evaluations indicates a great deal of room for improvement in this space. We leave further experimentation with LLM tutoring in TutorGym to future work. However, we must caution that running these evaluations (including those in Section 4) cost more than $730 US dollars in API fees to OpenAI and Anthropic, highlighting that inference costs are still a significant limitation to LLM-based tutoring approaches. We were able to run DeepSeek locally using available hardware.

4 Simulated Learner Evaluation

4.1 Method

We evaluate the Haiku-3.5 and GPT-4o LLMs as a simulated learner by running them through TutorGym’s interactive training pipeline with the Apprentice Tutors. We selected these two models because they were the most cost-effective within their performance class. We omitted DeepSeek-v2.5 because inference speed was prohibitively slow on our available hardware. However, we hope to investigate the use of open-source LLMs as simulated learners in future work.

To simulate tutored learning, we used a prompt similar to the one used for generating demos in the previous tutoring evaluation (see section 3.1). The main difference is that when the agent receives feedback from the tutor, it stores the `state`, `action`, and `correctness` values and adds these as in-context examples to

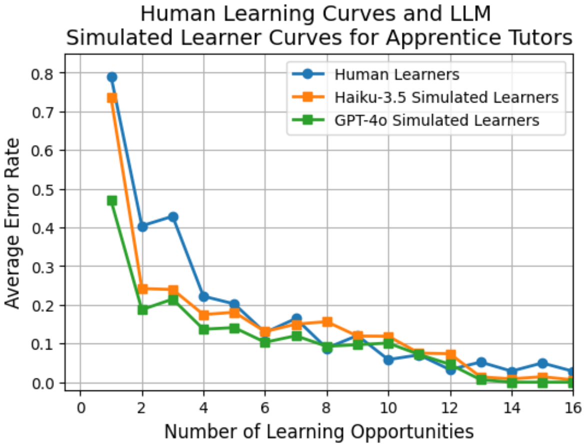


Fig. 3. Learning curves from humans as well as Haiku-3.5 and GPT-4o agents with in-context learning solving problems in Apprentice Tutors.

the prompt. The previously described tutoring evaluation prompt is essentially a zero-shot version of the prompt employed here. As the LLM agent experiences more problems, it accumulates an increasing number of in-context examples and the prompt size grows. Given the high cost of large prompts and the difficulty of locally running open-source LLMs that support large contexts, we limited the prompt length to 50k characters (approx. 15k tokens using the GPT-4o tokenizer). To keep prompts to this size, we removed the oldest in-context examples whenever the length limit was exceeded.[5] With this limit, the prompt could contain around 20-30 example states with their associated correct and incorrect actions, depending on the size of the states. The training sequence was not shuffled, and progressed one domain at a time, so previous training experiences from the same domain were usually retained within the context window.

To test this approach, we applied it to simulate learning with the Apprentice Tutors. We focused on this platform because we had access to human data for it, which made it possible to generate human learning curves. For each of the 30 domains, we trained the agents on the maximum number of problems solved by more than 4 of the 192 human learners in our data. This amounted to about 3 to 10 problem instances per domain. We ran five agent simulations with each LLM to generate simulated learning curves.

4.2 Results

Figure 3 shows the human and LLM agent learning curves. These curves reflect the error rates on the first attempt applying each skill within each problem, following the convention used by PSLC DataShop [14]. We associate each unique

[5] We chose this limit to make it feasible to compare with open-source models on our available hardware in future work.

tutor interface element with a single skill and the curves average over all skills across the 30 tutors. Remarkably, the agents' error rates are qualitatively quite similar to the average human student error rate.

4.3 Discussion

It should be emphasized that although "learning" is in the phrase *in-context learning*, it is arguably not a form of learning because it does not produce a permanent change in the model. The LLM's behavior is the product of a costly data-driven training process that includes more math problems than a human would solve in their lifetime. The in-context examples guide the LLM in applying this knowledge within the tutor. We hypothesize that they primarily provide information on the location and order of answer entry, rather than enabling the LLM to "learn" new math skills.

These results suggest that in-context learning may improve LLM tutoring capabilities over the zero-shot prompt we employed in section 3. However, we only evaluate whether the agent can generate a single correct action here, not its ability to grade complete action profiles.

For the purposes of illustrating how TutorGym can be used to facilitate the development of LLM-based simulated learners, this is a very interesting preliminary result. Before this demonstration, theory-driven computational models of learning [48,19] have been the only variety of simulated learners to produce human-like learning curves within ITSs. Reinforcement learning agents, for instance, show orders of magnitude slower learning [20]. LLMs, no doubt, have a great deal to offer in this space. We hope that TutorGym will become a valuable tool for AIED researchers to further these developments.

5 Conclusions & Future Work

TutorGym provides exciting new opportunities within AIED. LLM benchmarks typically lack the adaptive supports of ITSs, and prior tools for training simulated learners have been largely closed-source and limited to just a handful of domains. TutorGym offers tools for AI agents to interact with a growing suite of tutor domains (223 so far). We have illustrated how TutorGym can be used to evaluate AI agents as both tutors and as simulated learners.

TutorGym has potentially many more applications than we have shown. For instance, TutorGym could be used to evaluate how effectively simulated learner-based authoring tools like AI2T [47] are at learning ITS behavior from the author's instruction. In addition, researchers may use the hints available in its suite of tutors to evaluate automated hint generators. The possibilities are vast. TutorGym enables ITSs to be used not just as student-facing learning material, but also as environments for simulation and agent development. We look forward to seeing how others come to utilize it.

Acknowledgments Funded in part by NSF #2247790 and ARL #W911NF2120101, #W911NF2320203. Views are the author's and do not reflect those of the U.S. Govt.

References

1. Aleven, V., McLaren, B.M., Sewall, J., Koedinger, K.R.: The cognitive tutor authoring tools (CTAT): Preliminary evaluation of efficiency gains. In: International Conference on Intelligent Tutoring Systems. pp. 61–70. Springer (2006)
2. Aleven, V., McLaren, B.M., Sewall, J., Van Velsen, M., Popescu, O., Demi, S., Ringenberg, M., Koedinger, K.R.: Example-tracing tutors: Intelligent tutor development for non-programmers. International Journal of Artificial Intelligence in Education **26**(1), 224–269 (2016)
3. Anderson, J.R., Boyle, C.F., Corbett, A.T., Lewis, M.W.: Cognitive modeling and intelligent tutoring. Artificial intelligence **42**(1), 7–49 (1990)
4. Anderson, J.R., Schunn, C.D.: Implications of the ACT-R learning theory: No magic bullets. In: Advances in instructional Psychology, Volume 5, pp. 1–33. Routledge (2013)
5. Brockman, G.: OpenAI Gym. arXiv preprint arXiv:1606.01540 (2016)
6. Calo, T., Maclellan, C.: Towards educator-driven tutor authoring: generative AI approaches for creating intelligent tutor interfaces. In: Proceedings of the Eleventh ACM Conference on Learning@ Scale. pp. 305–309 (2024)
7. Cobbe, K., Kosaraju, V., Bavarian, M., Chen, M., Jun, H., Kaiser, L., Plappert, M., Tworek, J., Hilton, J., Nakano, R., et al.: Training verifiers to solve math word problems, 2021. URL https://arxiv. org/abs/2110.14168 (2021)
8. Dong, Q., Li, L., Dai, D., Zheng, C., Ma, J., Li, R., Xia, H., Xu, J., Wu, Z., Liu, T., et al.: A survey on in-context learning. arXiv preprint arXiv:2301.00234 (2022)
9. Graesser, A.C., Lu, S., Jackson, G.T., Mitchell, H.H., Ventura, M., Olney, A., Louwerse, M.M.: AutoTutor: A tutor with dialogue in natural language. Behavior Research Methods, Instruments, & Computers **36**, 180–192 (2004)
10. Gupta, A., Reddig, J., Caló, T., Weitekamp, D., MacLellan, C.J.: Beyond final answers: Evaluating large language models for math tutoring. In: Proceedings of the 26th International Conference on Artificial Intelligence in Education (AIED) (2025), `https://doi.org/10.48550/arXiv.2503.16460`
11. Hendrycks, D., Burns, C., Kadavath, S., Arora, A., Basart, S., Tang, E., Song, D., Steinhardt, J.: Measuring mathematical problem solving with the math dataset. arXiv preprint arXiv:2103.03874 (2021)
12. Käser, T., Alexandron, G.: Simulated learners in educational technology: A systematic literature review and a Turing-like test. International Journal of Artificial Intelligence in Education **34**(2), 545–585 (2024)
13. Kestin, G., Miller, K., Klales, A., et al.: AI tutoring outperforms active learning. Preprint, Research Square (2024). `https://doi.org/10.21203/rs.3.rs-4243877/v1`
14. Koedinger, K.R., Baker, R.S., Cunningham, K., Skogsholm, A., Leber, B., Stamper, J.: A data repository for the EDM community: The PSLC datashop. Handbook of educational data mining **43**, 43–56 (2010)
15. Koedinger, K.R., Corbett, A.T., Perfetti, C.: The knowledge-learning-instruction framework: Bridging the science-practice chasm to enhance robust student learning. Cognitive Science **36**(5), 757–798 (2012)
16. Koedinger, K.R., Matsuda, N., MacLellan, C.J., McLaughlin, E.A.: Methods for evaluating simulated learners: Examples from simstudent. In: AIED Workshops (2015)
17. Kulik, J.A., Fletcher, J.: Effectiveness of intelligent tutoring systems: a meta-analytic review. Review of Educational Research **86**(1), 42–78 (2016)

18. Liu, R., Koedinger, K.R.: Closing the loop: Automated data-driven cognitive model discoveries lead to improved instruction and learning gains. Journal of Educational Data Mining **9**(1), 25–41 (2017)
19. MacLellan, C.J.: Model human learners: Computational models to guide instructional design. In: Proceedings of the 47th Annual Meeting of the Cognitive Science Society. Cognitive Science Society, San Francisco, CA (2025)
20. MacLellan, C.J., Gupta, A.: Learning expert models for educationally relevant tasks using reinforcement learning. International Educational Data Mining Society (2021)
21. Maclellan, C.J., Harpstead, E., Patel, R., Koedinger, K.R.: The Apprentice Learner Architecture: Closing the loop between learning theory and educational data. International Educational Data Mining Society (2016)
22. MacLellan, C.J., Koedinger, K.R.: Domain-general tutor authoring with apprentice learner models. International Journal of Artificial Intelligence in Education pp. 1–42 (2020)
23. MacLellan, C.J., Stowers, K., Brady, L.: Evaluating alternative training interventions using personalized computational models of learning. Advances in Cognitive Systems **10**, 1–18 (2023)
24. Matsuda, N., Cohen, W.W., Koedinger, K.R.: Teaching the teacher: Tutoring SimStudent leads to more effective cognitive tutor authoring. International Journal of Artificial Intelligence in Education **25**(1), 1–34 (2015)
25. Mirzadeh, I., Alizadeh, K., Shahrokhi, H., Tuzel, O., Bengio, S., Farajtabar, M.: GSM-symbolic: Understanding the limitations of mathematical reasoning in large language models. arXiv preprint arXiv:2410.05229 (2024)
26. Mitrovic, A.: Modeling domains and students with constraint-based modeling. In: Advances in intelligent tutoring systems, pp. 63–80. Springer (2010)
27. Nwana, H.S.: Intelligent tutoring systems: an overview. Artificial Intelligence Review **4**(4), 251–277 (1990)
28. Pal Chowdhury, S., Zouhar, V., Sachan, M.: Autotutor meets large language models: A language model tutor with rich pedagogy and guardrails. In: Proceedings of the Eleventh ACM Conference on Learning@ Scale. pp. 5–15 (2024)
29. Pardos, Z.A., Bhandari, S.: ChatGPT-generated help produces learning gains equivalent to human tutor-authored help on mathematics skills. PLOS One **19**(5), e0304013 (2024)
30. Pardos, Z.A., Tang, M., Anastasopoulos, I., Sheel, S.K., Zhang, E.: OATutor: An open-source adaptive tutoring system and curated content library for learning sciences research. In: Proceedings of the 2023 CHI conference on human factors in computing systems. pp. 1–17 (2023)
31. Rachatasumrit, N., Carvalho, P.F., Li, S., Koedinger, K.R.: Content matters: A computational investigation into the effectiveness of retrieval practice and worked examples. In: International Conference on Artificial Intelligence in Education. pp. 54–65. Springer (2023)
32. Radmehr, B., Singla, A., Käser, T.: Towards generalizable agents in text-based educational environments: A study of integrating rl with LLMs. arXiv preprint arXiv:2404.18978 (2024)
33. Roll, I., Briseño, A., Yee, N., Welsh, A.: Not a magic bullet: The effect of scaffolding on knowledge and attitudes in online simulations. In: Proceedings of the International Conference of the Learning Sciences (ICLS). pp. 879–886. International Society of the Learning Sciences, Boulder, CO (2014), `https://repository.isls.org//handle/1/1208`

34. Schmucker, R., Xia, M., Azaria, A., Mitchell, T.: Ruffle&Riley: Insights from designing and evaluating a large language model-based conversational tutoring system. In: International Conference on Artificial Intelligence in Education. pp. 75–90. Springer (2024)
35. Self, J.: Computational mathetics: towards a science of learning systems design (1995)
36. Siddiqui, M.N., Gupta, A., Reddig, J.M., MacLellan, C.J.: HTN-based tutors: A new intelligent tutoring framework based on hierarchical task networks. In: Proceedings of the Eleventh ACM Conference on Learning @ Scale. p. 491–495 (2024)
37. Smith, G., Gupta, A., MacLellan, C.: Apprentice tutor builder: A platform for users to create and personalize intelligent tutors. arXiv preprint arXiv:2404.07883 (2024)
38. Stamper, J., Xiao, R., Hou, X.: Enhancing LLM-based feedback: Insights from intelligent tutoring systems and the learning sciences. In: International Conference on Artificial Intelligence in Education. pp. 32–43. Springer (2024)
39. Ur, S., VanLehn, K.: STEPS: a simulated, tutorable physics student. Journal of Artificial Intelligence in Education **6**, 405–437 (1995)
40. VanLehn, K.: Learning one subprocedure per lesson. Artificial Intelligence **31**(1), 1–40 (1987)
41. VanLehn, K.: Mind bugs: The origins of procedural misconceptions. MIT press (1990)
42. VanLehn, K.: Rule-learning events in the acquisition of a complex skill: An evaluation of cascade. The Journal of the Learning Sciences **8**(1), 71–125 (1999)
43. VanLehn, K.: The behavior of tutoring systems. International Journal of Artificial Intelligence in Education **16**(3), 227–265 (2006)
44. VanLehn, K.: The relative effectiveness of human tutoring, intelligent tutoring systems, and other tutoring systems. Educational Psychologist **46**(4), 197–221 (2011)
45. VanLehn, K., Ohlsson, S., Nason, R.: Applications of simulated students: An exploration. Journal of Artificial Intelligence in Education **5**, 135–135 (1994)
46. Weitekamp, D., Harpstead, E., Koedinger, K.: An interaction design for machine teaching to develop AI tutors. CHI (2020)
47. Weitekamp, D., Harpstead, E., Koedinger, K.: AI2T: Building trustable AI tutors by interactively teaching a self-aware learning agent. arXiv preprint arXiv:2411.17924 (2024)
48. Weitekamp, D., Koedinger, K.: Computational models of learning: Deepening care and carefulness in AI in education. In: International Conference on Artificial Intelligence in Education. pp. 13–25. Springer (2023)
49. Weitekamp, D., MacLellan, C.: Decomposed inductive procedure learning: Learning academic tasks with human-like data efficiency. In: Proceedings of the 47th Annual Meeting of the Cognitive Science Society. Cognitive Science Society, San Francisco, CA (2025)
50. Weitekamp, D., Ye, Z., Rachatasumrit, N., Harpstead, E., Koedinger, K.: Investigating differential error types between human and simulated learners. In: International Conference on Artificial Intelligence in Education. pp. 586–597. Springer (2020)
51. Weitekamp III, D., Harpstead, E., MacLellan, C.J., Rachatasumrit, N., Koedinger, K.R.: Toward near zero-parameter prediction using a computational model of student learning. International Educational Data Mining Society (2019)