ACTIVE EXPLORATION, RULE DISCOVERY, AND TEACHING IN ARC

Anonymous authors
Paper under double-blind review

ABSTRACT

Abstract reasoning benchmarks such as the Abstraction and Reasoning Corpus (ARC) typically provide models with a small set of demonstrations and evaluate whether they can infer the underlying transformation rule. However, the selection of these demonstrations and its role in successful inferences has received comparatively little attention. In this work, we instantiate interactive environments for ARC tasks using task-specific generator and verifier programs, allowing us to study both active discovery and teaching. We introduce ActiveARC, a framework in which models can construct input grids, observe their transformations through an oracle, and then solve held-out instances. We further ask whether models that already know the underlying rule can select demonstrations that effectively teach it to another model. Across four ARC-style datasets and five model families, requiring models to actively select their own examples substantially reduces performance relative to randomly sampled examples under matched query budgets. Accuracy drops by 12.8 percentage points for official items on average and 13.6 points for freshly sampled items, with active selection underperforming random sampling in all evaluated settings. We find that models systematically construct smaller experiments than both human-authored training examples and random samples from the task generator, and in most settings less diverse ones. Surprisingly, human-authored demonstrations often provide only a modest advantage over random generator samples, especially on freshly sampled items. In the teaching setting, when given explicit knowledge of the transformation, the strongest model generates demonstrations that teach other models nearly as effectively as human-authored examples. The other models teach considerably less effectively, even when they can apply the provided rules themselves. With an unconstrained interaction budget, most models do not discover substantially better and typically stop after only a few queries. Our results suggest that inferring a rule from given examples and gathering the right examples oneself are distinct capabilities, and that current models may lack the latter even when they have the former.

1 INTRODUCTION

Large language models have become increasingly capable at abstract reasoning tasks, including problems that require inferring latent rules from only a small number of examples. The Abstraction and Reasoning Corpus (ARC; (Chollet, 2019)) has become a particularly prominent test bed for studying this ability. ARC tasks consist of a small set of input-output demonstrations and ask the solver to infer an underlying transformation rule, which then has to be applied to a new input. Recent work has substantially improved model performance on ARC and related benchmarks, but almost all of this work assumes that the demonstrations themselves are fixed and given. The model's task is therefore one of passive induction: it must reason from whatever evidence it has been given. This setup leaves out an important part of intelligent reasoning. When trying to understand an unfamiliar system, an agent is often able to choose what evidence to collect. A scientist can design an experiment, a programmer can probe the behavior of an unknown function, and a learner can ask for an example that distinguishes between competing hypotheses. The usefulness of such interactions depends not only on the ability to infer a rule from evidence, but also on the ability to identify which evidence would be informative in the first place. There are notable differences between these

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

two abilities and strong inference does not necessarily imply strong evidence selection capability. A model may be able to recognize the correct transformation once it is shown a suitable set of examples while still being poor at constructing experiments that would reveal that transformation.

ARC provides a natural setting in which to separate these capabilities. Because tasks specify compact transformation rules over discrete grids, and recent programmatic reconstructions (Moffitt, 2025; Hodel, 2024; Anonymous, 2027) of ARC-style tasks make it possible to generate new inputs and compute their corresponding outputs. We use these programs to turn otherwise static ARC problems into interactive environments. Rather than receiving a fixed set of demonstrations, a model can construct an input grid, submit it to an oracle implementing the hidden transformation, observe the resulting output, and use this interaction to decide what to query next. After a fixed interaction budget, the model is evaluated on a held-out instance in the usual way. We call this setting *active discovery*, closely related to the established research field of active learning. It allows us to ask several interesting research questions: given the same number of demonstrations, does allowing a model to choose which inputs it observes actually help it infer the underlying rule? What if there is no external limit to demonstration selection? How diverse are model selected samples and is there an overarching strategy? To investigate these questions, we compare model-selected demonstrations against randomly sampled demonstrations from the same task generator, keeping the amount of available evidence matched. Perhaps surprisingly, we find that current models generally perform substantially worse when selecting their own evidence. Their experiments tend to cover a narrower portion of the task distribution, are typically smaller and often less diverse than the examples obtained through random sampling. Increasing the interaction budget does not straightforwardly resolve this problem, as models frequently stop experimenting after only a small number of queries.

The same framework also lets us study the complementary problem of *teaching*. For the original tasks, demonstration pairs were presumably not selected randomly, but instead chosen to give a full and perhaps intuitive account of the underlying rule. It is unclear whether models are capable of this as well, and if yes, how their choice of demonstrations compares against human ones. To study this, the model is given the underlying transformation rule in different formats and must construct a small set of demonstrations for another unknown model. This setting removes uncertainty about the rule itself and asks whether a model that already knows the solution can select examples that communicate it effectively. Comparing discovery and teaching therefore lets us distinguish between two closely related abilities: choosing evidence that is useful for oneself while the rule is unknown, and constructing useful evidence for another learner once the rule is known. We find a substantial asymmetry between the two. While active discovery remains difficult, there are more encouraging results for active teaching. When able to reliably execute the provided rule, our strongest evaluated model is able to construct demonstrations that teach another model at a level comparable to human-authored examples. Together, these experiments complement the current function of ARC as a passive few-shot induction problem by the challenge of actively acquiring informative evidence. With this new setting, we want to highlight a distinction that is largely absent from current abstract reasoning evaluation: reasoning over a given set of observations and deciding which observations are worth obtaining. ActiveARC makes this distinction explicit by turning ARC from a static induction benchmark into an environment for studying discovery and teaching.

2 Related Work

Connection to Active Learning and Machine Teaching. Active learning has been a longstanding research direction in machine learning (Cohn et al., 1994; Settles, 2009). Classical active learning typically considers a supervised learning setting in which a learner selects informative examples to improve its predictions while minimizing labeling cost. Our active discovery setting is closest to the direction of query synthesis, but differs in the structure of the inference problem: the model constructs a small number of structured input grids, observes their transformations under an unknown rule, and uses these observations to update its hypothesis about that rule. The complementary problem of selecting examples to facilitate another learner's inference has been studied in machine teaching and computational models of pedagogy (Zhu, 2015; Shafto et al., 2014). In our teaching setting, a teacher with access to the underlying transformation rule constructs demonstrations intended to help a separate pretrained reasoning model infer that rule and solve a held-out instance. ActiveARC therefore studies both sides of example selection in abstract rule induction: how a learner chooses evidence to discover a rule and how a knowledgeable teacher chooses evidence to communicate it.

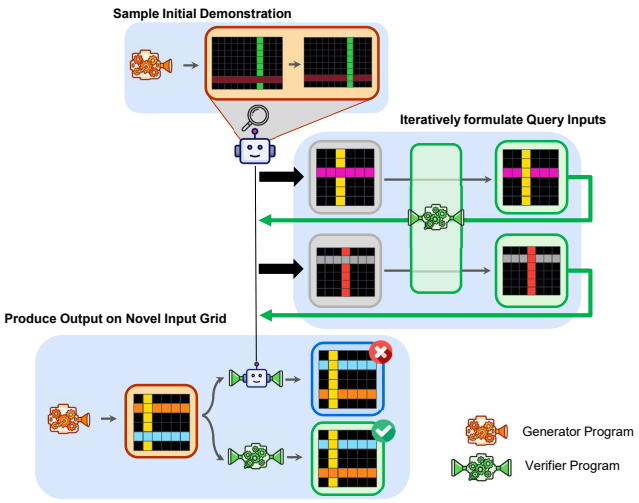

Figure 1: Overview of the active discovery setting. The task's generator program samples an initial input-output demonstration (top). The model then iteratively formulates query inputs, each of which is transformed by the verifier program and returned to the model as an additional input-output pair (middle), either for a fixed budget of K queries (Active-K) or until the model decides to request the test (Active-Free). Finally, the model has to produce the output for a novel input grid sampled from the generator (as well as for the official test inputs), which is scored by exact match against the output of the verifier (bottom).

Active Learning in Abstract Reasoning. Recent work has begun to study active experimentation in abstract reasoning. ZendoWorld (Koehler et al., 2026) evaluates active visual concept induction, while Piriyakulkij et al. (2024) study experiment selection and online hypothesis revision in Zendo-style and ActiveACRE tasks. Relatedly, RULEARN (He et al., 2025) evaluates interactive rule learning in simulated environments. ActiveARC differs in studying structured transformations: each query returns an entire transformed output rather than a binary label, and we directly compare actively selected evidence against matched random demonstrations. We additionally consider the complementary problem of selecting demonstrations to teach a rule to another model.

Generative and Interactive ARC Tasks. ARC was originally released as a fixed collection of input–output demonstrations, but recent work has expanded the domain through procedural generation. RE-ARC (Hodel, 2024) and ARC-GEN (Moffitt, 2025) provide task-specific generators for producing new instances of ARC tasks, while concurrent work extends procedural generation to ConceptARC (Anonymous, 2027). Such generators have primarily been used to expand evaluation and training data; for example, PotARCin (Beger et al., 2026b) uses programmatically generated instances to evaluate several dimensions of rule understanding beyond standard ARC accuracy. ActiveARC instead uses these generators as interactive environments, allowing models to construct whichever inputs they want to observe transformed. ARC-AGI-3 (Foundation, 2026) likewise makes abstract reasoning interactive, but through game-like environments requiring exploration, goal discovery, and planning rather than active experimentation over ARC-style transformations.

3 Methodology

3.1 Task Sources

In order to instantiate an appropriate environment for the active discovery setting, we first consider a set of different ARC datasets. The ARC-AGI-1 training set is a natural first candidate, since it has been a standard benchmark for evaluating abstract reasoning in the ARC domain and provides a widely studied reference point for comparing model performance. Just as importantly for our setting, both RE-ARC (Hodel, 2024) and ARC-GEN (Moffitt, 2025) provide task-specific procedural generators and verifiers for these tasks, allowing us to sample new inputs from the same underlying transformations. More recently, Moffitt (2025) added implementations for a subset of ARC-AGI-

2 training-set tasks, a more challenging follow-up to the original benchmark. We independently source additional verifiers from three public GitHub repositories (GitMonsters, 2026; Pang, 2025; Sehrawat, 2026) and validate them against the official training and test examples. Each verifier is additionally stress-tested on 3,000 samples from the corresponding task generator to ensure agreement, yielding 200 tasks for which both a valid verifier and generator are available. We further validate the generators by manually reviewing all verifier–generator disagreements and inspecting the underlying programs to ensure that the official training and test examples fall within the generator's sampling distribution. Across both benchmarks, we explicitly modify 111 generators whose original sampling distributions did not adequately cover the official examples. Further, we include ConceptARC (Moskvichev et al., 2023), a set of 160 tasks centered around 16 core concepts, for which Anonymous (2027) recently proposed a DSL-based implementation of generators and verifiers. ConceptARC is further motivated as a domain by Beger et al. (2026a), who find that a large share of correct model solutions may actually be a form of programmatic shortcuts. If that is the case, active discovery may further highlight this misalignment (since queries would center around fundamentally incorrect solution proposals). Finally, we utilize the recently proposed P-ARC dataset from Beger et al. (2026b), which is a set of 50 tasks with hand-engineered generator and verifier programs. This dataset is especially interesting because of its elevated difficulty and the low likelihood of potential data contamination due to its recent release date. In general, it is likely that evaluated models were trained on the selected datasets, which is why we include an additional test input which is sampled randomly from the generator, and exclude official samples in all settings except for the static baseline one. Correspondingly, it is quite unlikely that models would be able to use (approximate) retrieval/recall to guess the output, although we cannot guarantee that the rules themselves were not previously acquired.

3.2 Active Discovery

Using these paired verifiers and generators, we instantiate a task "environment" (Figure 1), where an agent can freely synthesize an input grid to query the underlying verifier with. After exhausting the given budget or by its own admission in the free setting, the agent then enters a testing stage in which it has to apply the learned transformation to one or more new inputs. Each trial starts with a single input-output pair. Without it, the agent would have to guess even the most basic properties of the task (such as the grid dimensions or the colors involved), and most initial queries would be spent just on establishing what kind of input the transformation expects. The initial sample is drawn from the task generator rather than taken from the official training examples, so that the agent never sees an authored demonstration. From there, the agent can construct an input grid (exchanged as a JSON list of integers between 0 and 9) and submit it to the environment, which runs the verifier on it and returns the corresponding output grid. Not every grid is a valid input for every transformation, so if the verifier fails on a submitted grid, the agent is told that the input is invalid or the rule is not applicable, and the query is not counted. We consider two versions of this setting. In the *Active-K* setting, the agent has to submit exactly K valid queries before being allowed to request the test, where K is the number of official training pairs minus one. Together with the initial sample, the agent therefore ends up with exactly as many input-output pairs as the original task provides, which makes this setting directly comparable to the baselines described below. The agent is explicitly told that queries do not cost anything and that there is no credit for using fewer of them. In the *Active-Free* setting, there is no query budget at all and the agent decides by itself when it has seen enough and wants to be tested (only limited by a generous cap of 64 interaction turns). In this case, the agent is told that its answer is scored and that the number of submitted queries will be recorded, but not that fewer queries are preferable, since we do not want to discourage experimentation either way.

3.3 Baselines and Evaluation

To put the active settings into perspective, we compare them against two passive baselines, in which the agent receives a fixed set of demonstrations and is directly tested afterwards. In the *static* setting, the agent receives the official training pairs of the task, which corresponds to the standard ARC evaluation. In the *random-K* setting, the agent instead receives the same kind of generator-drawn initial sample followed by further random samples from the task generator, again resulting in the same number of pairs as the official task (and as the Active-K setting). Random-K is therefore the most direct reference point for active discovery: the amount of evidence is identical (e.g., no potential bias towards the test queries), and the only difference is whether the agent chose the inputs

itself or whether they were sampled from the generator. The static setting in turn tells us whether human-authored demonstrations carry more information than random samples of the same size. In all settings, the testing stage is identical. The agent is presented with the official test input(s) of the task as well as one additional test input that is freshly sampled from the generator. As discussed above, the latter guards against models simply recalling the official solution, and comparing the two gives us some indication of how much performance on the official items is dependent on familiarity with the task. Each test input is answered in a separate branch that contains the full interaction history (or the given demonstrations) up to that point, but none of the other test inputs, so that solving one item cannot inform the answer to another. The agent is not told whether its answer was correct, and each answer is scored by exact match with the verifier output. We report accuracy separately on the official items (where a trial only counts as correct if all official test inputs are solved) and on the sampled items. Trials in which the agent never reaches the testing stage (for instance because it runs out of turns or keeps producing reasoning without ever calling a tool) are counted as incorrect. We evaluate five models from different providers: GPT-6 Astra, GPT-5.6 Luna, Claude Opus 5, Gemini 3.8 Flash, and DeepSeek V4.1 Flash. All models are run with low reasoning effort, which keeps the cost of several thousand multi-turn trials manageable, and are accessed through their respective first-party APIs where available.

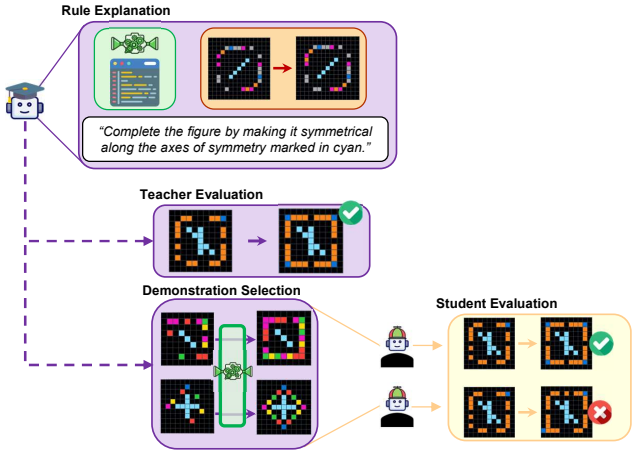

Figure 2: Overview of the teaching setting. The teacher receives the rule as a natural-language description, the source code of the verifier, and one example pair (Rule Explanation). Before teaching, it is evaluated on the test inputs itself, in a separate context that is discarded afterwards (Teacher Evaluation). It then constructs demonstration inputs, whose outputs are computed by the verifier, so that all demonstrations are correct by construction (Demonstration Selection). Finally, the demonstrations replace the official training pairs for a separate student model, which is evaluated on the same test inputs (Student Evaluation).

3.4 Teaching

In the Teaching setting (Figure 2), the designated teacher model receives a set of inputs to establish an initial understanding of the underlying rule. These consist of a natural-language description of the rule together with the source code of the verifier (for DSL-based verifiers we synthesize the underlying code accordingly), as well as one example pair. Rules for ARC-AGI-1 and ARC-AGI-2 are taken from validated descriptions in LARC (Acquaviva et al., 2022, 364 tasks) and MARC2 (Baumgaertner, 2026, 168 tasks); the remaining 68 are written and validated by us, and those for P-ARC and ConceptARC were provided by the dataset authors through personal correspondence. It then designs input grids, and the environment computes the corresponding outputs with the verifier. This way, the demonstrations are always correct by construction, and the teaching quality depends only on which inputs the teacher decides to show, not on whether it can execute the rule correctly. The teacher is given the same budget as the original task (i.e., as many demonstrations as there are official training pairs) and cannot see how the student responds to them. Before teaching, we additionally test each teacher on the same test inputs as the student, which lets us relate teaching quality to how well the teacher can apply the rule itself. The student is then evaluated exactly as

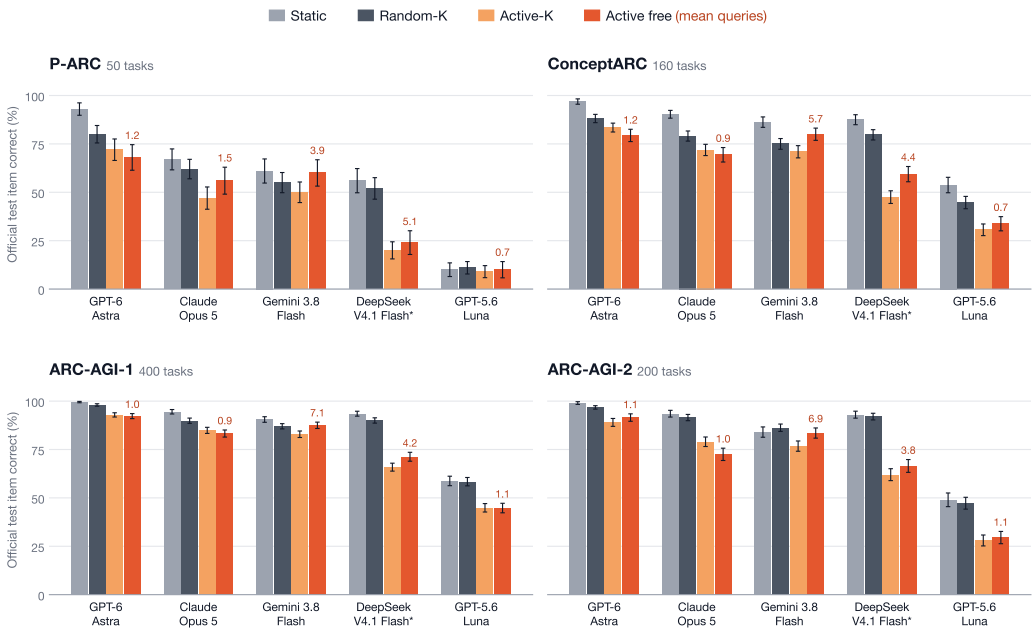

Figure 3: Accuracy on the official test items for the two passive baselines (static, random-K) and the two active settings (Active-K, Active-Free) across all five models and four benchmarks. Random-K and Active-K provide exactly as many input-output pairs as the official task, while Active-Free imposes no query budget; the number above each Active-Free bar gives the mean number of queries per task. Error bars show ± 1 standard deviation of accuracy over tasks (bootstrap). Static, Random-K and Active-K are averaged over two independent draws and Active-Free is a single run due to the strongly increased budget requirements. *DeepSeek V4.1 Flash abandons up to 44% of active trials before reaching the testing stage; these trials are counted as incorrect (see Appendix A.3 for results excluding them).

in the static baseline, with the teacher-constructed demonstrations taking the place of the official training pairs. In addition, we evaluate the recently proposed VARC model (Hu et al., 2025), which is based on the Vision-Transformer architecture, as a non-LLM student. Starting from an untrained model, VARC is trained from scratch on each task independently, using only the demonstrations of that task (either the official training pairs or the teacher-constructed ones) together with augmented variants of them, and is then evaluated on the test inputs of the task (pass@2). Since it cannot draw on any prior knowledge, it provides a baseline for teaching efficiency that depends entirely on the demonstrations.

4 Results

4.1 Active Discovery

Figure 3 shows our results on the active discovery setting (see Table 1 for the underlying numbers). When comparing passive and active sample selection, it is evident that requiring models to choose their own inputs consistently hurts performance: Active-K falls below random-K in all 20 combinations of model and benchmark, by 12.8 percentage points on average on the official items (ranging from 2.0 to 32.2 points) and by 13.6 points on the sampled items (see Appendix A.1). Since both settings provide exactly the same number of input-output pairs and start from the same kind of initial sample, the difference can only come from which inputs end up being observed. The size of the gap varies considerably between models. It is smallest for GPT-6 Astra (6.4 points) and Gemini 3.8 Flash (5.7 points) and largest for DeepSeek V4.1 Flash (29.6 points), although a substantial part of the latter is due to trials that DeepSeek never completes (between 10 and 56 trials per run, which we count as incorrect). Across benchmarks, the gap is largest on ARC-AGI-2 (15.8 points), where

GPT-6 Astra and Claude Opus 5 solve over 90% of the tasks from random samples but lose around 8 and 13 points, respectively, once they have to select the inputs themselves. The comparison between the two passive baselines is interesting as well. On the official items, the human-authored demonstrations provide an advantage of 4.7 points over random samples, and random-K matches or exceeds static in only 2 of 20 cells. On the sampled items, however, the bulk of this advantage disappears (0.3 points on average), and random-K matches or exceeds static in half of all cells. This suggests that the benefit of the official demonstrations is at least partly tied to the official test items themselves, either because they were authored together or because of familiarity with the original task, rather than to a generally more informative choice of examples. Finally, removing the query budget does not resolve the problem. Active-Free improves on Active-K by only 2.7 points on average and remains 10.1 points below random-K. The one notable exception is Gemini 3.8 Flash, which matches or exceeds random-K on three of four benchmarks in the free setting, and, as we discuss next, is also the model that makes the most use of the additional freedom.

4.2 What Models Ask For

To better understand where the gap comes from, we compare the inputs that models construct in the Active-K setting with the random generator samples of the random-K setting, matched in number. The most consistent difference is size: model-constructed inputs only cover about half the area of random samples (84 compared to 174 cells) on average, and they are smaller in all 20 cells. They are also smaller than the human-authored training inputs in 19 of 20 cells (84 compared to 135 cells on average), with DeepSeek V4.1 Flash on P-ARC as the only exception (Table 7). In addition, they are frequently less diverse. In terms of the mean pairwise distance between the inputs of a trial, model-constructed evidence is less diverse than random samples in 12 of 20 cells, including every model on P-ARC and ConceptARC, and around 15% of all queries are near-copies of the initial sample (at most 10% of cells changed), compared to 4% of random samples. Taken together, models tend to probe the transformation with small, simple inputs close to what they have already seen, rather than with the kind of rich and varied instances that the task generators (and the human authors) produce. As for an overarching query strategy, we find no systematic experimentation in the sense of deliberately covering the factors of variation of a task. Instead, the dominant pattern appears to be the construction of simplified variants of the initial sample that test a hypothesis already formed from it, often leaving alternatives untested, while a systematic coverage of the relevant factors occurs only for individual tasks (see Appendix A.15 for examples). The query counts in the Active-Free setting point in the same direction. GPT-6 Astra, Claude Opus 5 and GPT-5.6 Luna submit only about one query per task on average and do not query at all in 30 to 53% of tasks, i.e., they mostly go straight to the test after seeing the initial sample (Figure 6). Gemini 3.8 Flash and DeepSeek V4.1 Flash query considerably more (3.8 to 7.1 queries per task), and Gemini is correspondingly the only model whose free-setting accuracy approaches or exceeds random-K. To separate the quality of the gathered evidence from the inference itself, we additionally let GPT-6 Astra answer the P-ARC test items from the evidence that other models gathered in the Active-K setting. With only the initial sample, it solves between 60 and 67% of the official items, and adding the respective model's queries changes this by at most three points (and by two points for its own queries), whereas the same number of random generator samples brings it to 80%. In other words, the actively chosen queries add very little information beyond the initial sample, regardless of which model selected them (Table 9).

4.3 Teaching

Figure 4 shows the results of the teaching setting, in which each of the five models teaches each of the five models (including itself) on all four benchmarks. Here, the models seem to be able to select evidence much more efficiently, at least for the strongest model. Demonstrations constructed by GPT-6 Astra teach almost exactly as well as the human-authored ones: averaged over all students and benchmarks, students taught by Astra lose only 1.1 points compared to the static setting, and no single combination deviates by more than six points. All other teachers clearly fall short, with average differences of 8.3 (Claude Opus 5), 8.9 (Gemini 3.8 Flash), 9.9 (DeepSeek V4.1 Flash) and 16.9 points (GPT-5.6 Luna). The differences are largest on P-ARC and smallest on ARC-AGI-2, where most teachers stay within five points of the static setting. Teaching oneself does not help either: Claude Opus 5, for instance, solves 67% of P-ARC tasks from the official demonstrations but only 48% from its own. Since all demonstrations are computed by the verifier, these differences

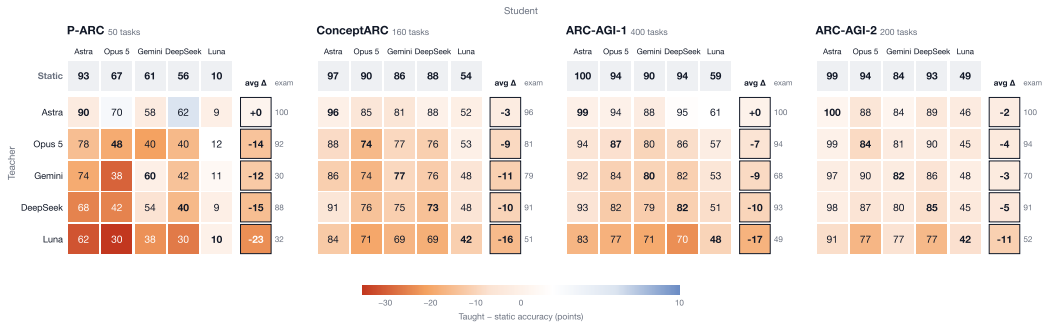

Figure 4: Teaching results on the official test items. Each cell gives the accuracy of a student (columns) taught with the demonstrations constructed by a teacher (rows) under the same budget as the official task, while the top row gives each student's static accuracy with the official training pairs. Cell colors encode the difference to the static accuracy, the framed column gives the teacher's mean difference over all five students, and the grey column the teacher's own accuracy on the test items before teaching ("exam"). Self-taught pairs are shown in bold, and teachers are sorted by their mean difference across benchmarks.

cannot stem from incorrect outputs, but only from the choice of inputs. The teachers' own test accuracy before teaching offers a partial explanation. GPT-5.6 Luna, the weakest teacher, also cannot reliably apply the rules it is given (32 to 52% on the official items), whereas GPT-6 Astra solves almost all of them. Being able to apply the rule is not sufficient, however: Claude Opus 5 and DeepSeek V4.1 Flash solve 81 to 94% of the items themselves and still teach considerably worse than Astra (see Appendix A.12 for an example).[1] An open question is how much of the teaching success can be attributed to the student rather than the teacher, since a capable student may be able to compensate for less informative demonstrations with its prior knowledge. The VARC model offers a useful point of reference here (Section 3). On all 400 ARC-AGI-1 tasks, VARC solves 39.5% of the tasks with the official demonstrations (pass@2), but only 31.0% and 29.2% with those from GPT-6 Astra and Claude Opus 5, and between 15.8 and 19.2% with those of the remaining teachers (Appendix A.14). All teachers remain significantly below the official demonstrations (by 8.5 to 23.8 points; 95% bootstrap confidence intervals over tasks exclude zero), and their ranking closely mirrors the one obtained with LLM students. Here, however, even Astra's demonstrations leave a significant gap to the human-authored ones. This suggests that even the best teacher demonstrations are less informative than the human-authored ones, and that strong LLM students are to some extent able to compensate for this.

4.4 Robustness

We run several additional analyses to test whether the gap between active and passive evidence could be explained by factors other than the choice of inputs itself. First, it is not driven by abandoned trials: if we exclude trials that never reach the testing stage instead of counting them as incorrect, the average gap between Active-K and random-K shrinks from 12.8 to 11.0 points on the official items (and from 13.6 to 11.5 points on the sampled items), with Active-K still below random-K in all 20 cells (Appendix A.3). Second, a slightly larger budget does not close it either. When we grant two additional queries on P-ARC (K+2), Claude Opus 5 improves by 11 points, but the other models change by at most four points, and all of them remain below random-K despite now observing two more input-output pairs (Appendix A.4). Third, the models do not adapt their experimentation when the initial sample is misleading. In a variant of the Active-Free setting on 230 ARC-AGI-1 tasks, which we call *slippage*, the initial sample is drawn from a narrower input distribution that is also consistent with a narrower (incorrect) rule, while queries are answered and test items are drawn under the broader original rule. Accuracy drops by 10 to 22 points compared to the sampled items of the regular Active-Free setting, while the number of queries remains essentially unchanged, i.e., the

[1] Gemini 3.8 Flash answers between 36 and 84% of these test items without any reasoning (despite the same reasoning setting as in all other experiments), and is considerably less accurate on these items (19 to 71%) than on the ones where it does reason (84 to 93%), so its test accuracy should be seen as a lower bound.

models rarely probe beyond what the initial sample suggests (Appendix A.5). Fourth, the models do adapt their experimentation to what they are asked to produce. When they have to submit a Python program that reproduces all held-out examples of the rule instead of a single output grid, GPT-6 Astra and Claude Opus 5 query about twice as often as in the regular Active-Free setting and skip querying entirely in far fewer tasks, although the absolute number of queries remains small (around 2.4 per task; Appendix A.6). Finally, one might argue that active discovery is simply fragile, in the sense that a single uninformative or misleading observation derails the agent. To test this, we repeat the Active-Free setting on ARC-AGI-1 for GPT-6 Astra and Gemini 3.8 Flash while corrupting a quarter of all query outputs (through mutated verifiers, color permutations, or outputs of a different task instance), without telling the agent which outputs are affected. Neither model is significantly affected: GPT-6 Astra solves 93.8% of the official items compared to 92.2% without noise, and Gemini 3.8 Flash 84.8% compared to 87.5%. Both models also appear to notice suspicious outputs, re-submitting the same input after 32% (Astra) and 42% (Gemini) of all corrupted outputs, compared to 2.4% and 8.8% of correct ones, and almost never in the noise-free setting (Appendix A.11). The failure of active discovery is therefore not one of being misled by what the environment returns, but of asking for too little in the first place.

5 Conclusion

Limitations. Certain limitations of our setup should be kept in mind when interpreting the results. First, the verifiers were written and validated for the input distribution of each task, but they also occasionally accept inputs (far) outside of it, for which the returned output may not reflect the intended rule and could therefore mislead the agent. We reject queries on which the verifier fails, but cannot guarantee that every accepted output matches what the original task authors would consider correct. Given that ARC tasks usually focus on a small and constrained rule domain, this is an innate drawback of the automated verifier setting. Second, all main experiments use low reasoning effort to keep the cost of several thousand multi-turn trials manageable. Higher effort only slightly increases querying for GPT-6 Astra on P-ARC (Appendix A.7), but other models or benchmarks may react differently. Third, ARC-AGI-1 and ARC-AGI-2 are close to saturated for the strongest models in the passive settings, which compresses the differences between conditions and makes these benchmarks more susceptible to contamination, a concern we can only partially address with freshly sampled test items and the more recent P-ARC. Finally, the Active-Free setting and several of its variants are single runs and model behavior may depend on the exact prompt, which we deliberately kept neutral rather than explicitly encouraging exploration.

In this work, we translated questions of active exploration, rule discovery and teaching to the previously static ARC domain. For that purpose, we introduced ActiveARC, a framework that turns ARC-style tasks into interactive environments by pairing task-specific generators and verifiers, and used it to study two sides of example selection in abstract rule induction: gathering evidence to discover an unknown rule, and constructing evidence to teach a known one. Our results show a clear gap between inferring a rule from given examples and choosing informative examples oneself. Across five models and four benchmarks, requiring models to select their own inputs consistently lowers accuracy compared to the same number of randomly sampled examples, by 12.8 points on the official and 13.6 points on freshly sampled test items. The inputs that models construct are smaller than both random samples and human-authored examples, frequently less diverse, and often close to what they have already seen, and when given an unconstrained budget, most models stop after one or two queries or none at all. Replaying the gathered evidence to a strong model confirms that these queries add little information beyond the initial sample, and the gap persists under a larger budget, noisy observations and when abandoned trials are excluded. Teaching turns out to be more tractable, at least for the strongest model, whose demonstrations teach other models almost as effectively as the human-authored ones, while all other teachers fall clearly short even when they can apply the rule themselves. Likewise, for a student trained from scratch, which cannot compensate for weak demonstrations with prior knowledge, even the best model-constructed demonstrations remain much less informative than those designed by humans. We hope that ActiveARC complements the view of ARC as a passive few-shot induction problem with the challenge of actively acquiring and communicating informative evidence, and that it provides a useful test bed for developing models that not only reason over the observations they are given, but also know which observations are worth obtaining.

Reproducibility Statement

We will make all data used in this work, including the generator and verifier programs of all four benchmarks, the sampled test items, and the complete interaction records of all experiments, as well as the code of the ActiveARC framework and our analysis scripts publicly available upon publication. The construction of the task environments, the active discovery and teaching settings, the baselines and the evaluation protocol are described in detail in Section 3. The prompts of the active and teaching settings are reproduced in Appendix A.16, the full numerical results in Appendix A.2, and the additional experimental variants in Appendices A.3 to A.11.

AI Use Statement

We acknowledge the use of generative AI tools for language revisions, such as grammar checks and reformulations. In addition, we used generative AI tools such as Claude Code to assist with code implementation, in particular for running well-defined experiments across different API endpoints, and to assist with the design of some of our plots. We reviewed all code generated with these tools and tested it for correctness. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

References

Samuel Acquaviva, Yewen Pu, Marta Kryven, Theodoros Sechopoulos, Catherine Wong, Gabrielle E. Ecanow, Maxwell Nye, Michael Henry Tessler, and Joshua B. Tenenbaum. Communicating natural programs to humans and machines. In *Advances in Neural Information Processing Systems*, volume 35, pp. 3731–3743, 2022.

Anonymous. Conceptarc at scale: Studying conceptual understanding in reasoning models. In *Submitted to International Conference on Learning Representations*, 2027. URL https://openreview.net/forum?id=2siSqdx8wm.

Bert Baumgaertner. Marc2: Metaphor abstraction and reasoning corpus v2, 2026. URL https://github.com/bertybaums/marc2.

Claas Beger, Ryan Yi, Shuhao Fu, Kaleda Denton, Arseny Moskvichev, Sarah W. Tsai, Sivasankaran Rajamanickam, and Melanie Mitchell. Do ai models perform human-like abstract reasoning across modalities?, 2026a. URL https://arxiv.org/abs/2510.02125.

Claas Beger, Ryan Yi, and Melanie Mitchell. Potarcin: Multi-dimensional evaluation of skill acquisition in abstract reasoning tasks, 2026b. URL https://arxiv.org/abs/2609.27288.

François Chollet. On the measure of intelligence, 2019. URL https://arxiv.org/abs/1911.01547.

David Cohn, Les Atlas, and Richard Ladner. Improving generalization with active learning. *Mach. Learn.*, 15(2):201–221, May 1994.

ARC Prize Foundation. Arc-agi-3: A new challenge for frontier agentic intelligence, 2026. URL https://arxiv.org/abs/2603.24621.

GitMonsters. SOLVED-562-verified: 562 verified ARC-AGI solvers. https://github.com/GitMonsters/SOLVED-562-verified, 2026. GitHub repository, accessed September 2026.

Kaiyu He, Mian Zhang, Shuo Yan, Peilin Wu, and Zhiyu Zoey Chen. IDEA: Enhancing the rule learning ability of large language model agent through induction, deduction, and abduction. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 13563–13597, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-256-5. doi: 10.18653/v1/2025.findings-acl.698. URL https://aclanthology.org/2025.findings-acl.698/.

Michael Hodel. Addressing the abstraction and reasoning corpus via procedural example generation, 2024. URL https://arxiv.org/abs/2404.07353.

Keya Hu, Ali Cy, Linlu Qiu, Xiaoman Delores Ding, Runqian Wang, Yeyin Eva Zhu, Jacob Andreas, and Kaiming He. Arc is a vision problem!, 2025. URL https://arxiv.org/abs/2511.14761.

Sophia Koehler, Antonia Wüst, Inga Ibs, Wasu Top Piriyakulkij, Wolfgang Stammer, Constantin Rothkopf, Kevin Ellis, and Kristian Kersting. Playing zendoworld: Challenging ai agents on active visual concept induction, 2026. URL https://arxiv.org/abs/2607.08233.

Michael D. Moffitt. Arc-gen: A mimetic procedural benchmark generator for the abstraction and reasoning corpus, 2025. URL https://arxiv.org/abs/2511.00162.

Arseny Moskvichev, Victor Vikram Odouard, and Melanie Mitchell. The ConceptARC benchmark: Evaluating understanding and generalization in the ARC domain. *Transactions on Machine Learning Research*, 2023.

Eric Pang. arc_agi: A state-of-the-art approach for ARC-AGI-2. https://github.com/epang080516/arc_agi, 2025. GitHub repository.

Wasu Top Piriyakulkij, Cassidy Langenfeld, Tuan Anh Le, and Kevin Ellis. Doing experiments and revising rules with natural language and probabilistic reasoning. In *Advances in Neural Information Processing Systems*, volume 37, pp. 53102–53137, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/5f1b350fc0c2affd56f465faa36be343-Abstract-Conference.html.

Arun Sehrawat. Verified python solutions for ARC-AGI-2. https://github.com/ArunSehrawat/arc-agi2-solutions, 2026. GitHub repository.

Burr Settles. Active learning literature survey. Computer Sciences Technical Report 1648, University of Wisconsin–Madison, 2009.

P. Shafto, N. D. Goodman, and T. L. Griffiths. A rational account of pedagogical reasoning: Teaching by, and learning from, examples. *Cognitive Psychology*, 71:55–89, 2014. doi: 10.1016/j.cogpsych.2013.12.004.

Xiaojin Zhu. Machine teaching: An inverse problem to machine learning and an approach toward optimal education. *Proc. Conf. AAAI Artif. Intell.*, 29(1), March 2015.

A Appendix

A.1 Results on Sampled Test Items

Figure 5 repeats the main comparison on the test inputs that were freshly sampled from the task generators. Active-K again falls below random-K in all 20 cells (by 13.6 points on average), while the advantage of the human-authored demonstrations over random samples largely disappears. Note that on P-ARC and ConceptARC, whose generators cannot be seeded, the sampled test input of the active settings is drawn anew in every trial, whereas the passive baselines and the teaching setting share one fixed sampled input per task. On these two benchmarks, the comparison is therefore between inputs from the same distribution rather than between identical inputs.

A.2 Detailed Results

Table 1 lists the accuracies underlying Figures 3 and 5.

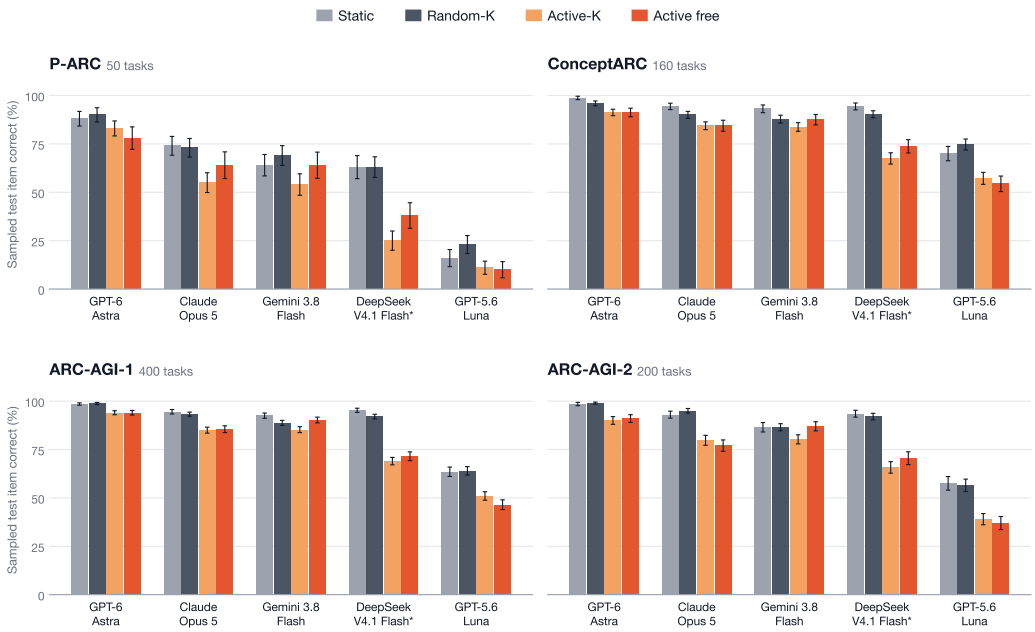

Figure 5: Accuracy on the freshly sampled test items, analogous to Figure 3. Error bars show ± 1 standard deviation of accuracy over tasks (bootstrap). *Abandoned active trials of DeepSeek V4.1 Flash are counted as incorrect.

		Official items				Sampled items			
Benchmark	Model	Static	Rand.-K	Act.-K	Act.-Free	Static	Rand.-K	Act.-K	Act.-Free
P-ARC	GPT-6 Astra	93.0	80.0	72.0	68.0	88.0	90.0	83.0	78.0
	Claude Opus 5	67.0	62.0	47.0	56.0	74.0	73.0	55.0	64.0
	Gemini 3.8 Flash	61.0	55.0	50.0	60.0	64.0	69.0	54.0	64.0
	DeepSeek V4.1 Flash*	56.0	52.0	20.0	24.0	63.0	63.0	25.0	38.0
	GPT-5.6 Luna	10.0	11.0	9.0	10.0	16.0	23.0	11.0	10.0
ConceptARC	GPT-6 Astra	96.9	88.1	83.4	79.4	98.8	95.9	91.2	91.2
	Claude Opus 5	90.3	79.1	71.9	69.4	94.4	90.0	84.4	84.4
	Gemini 3.8 Flash	86.2	75.0	70.9	80.0	93.1	87.8	83.8	87.5
	DeepSeek V4.1 Flash*	87.5	79.7	47.5	59.4	94.4	90.3	67.5	73.8
	GPT-5.6 Luna	53.8	44.7	30.6	33.8	70.0	74.7	57.2	54.4
ARC-AGI-1	GPT-6 Astra	99.5	98.0	92.9	92.2	98.5	98.8	94.0	94.0
	Claude Opus 5	94.5	89.9	84.9	83.2	94.5	93.2	85.0	85.5
	Gemini 3.8 Flash	90.5	87.0	82.9	87.5	92.5	88.8	85.2	90.2
	DeepSeek V4.1 Flash*	93.5	90.0	65.9	71.2	95.2	92.0	69.0	71.5
	GPT-5.6 Luna	58.8	58.4	44.9	44.8	63.5	64.0	51.0	46.5
ARC-AGI-2	GPT-6 Astra	99.0	96.8	89.0	91.5	98.5	99.0	90.0	91.0
	Claude Opus 5	93.5	91.5	79.0	72.5	93.0	95.0	79.8	77.0
	Gemini 3.8 Flash	84.0	86.2	76.8	83.5	86.5	86.5	80.2	87.0
	DeepSeek V4.1 Flash*	93.0	92.0	62.0	66.5	93.5	92.0	65.8	70.5
	GPT-5.6 Luna	49.0	47.2	28.0	29.5	57.5	56.5	39.0	37.0

Table 1: Accuracy (%) underlying Figures 3 and 5. Random-K and Active-K are averaged over two independent draws, static over the available runs, and Active-Free is a single run. *Abandoned trials are counted as incorrect (see Table 2).

A.3 Abandoned Trials

Throughout the paper, trials in which the agent never reaches the testing stage are counted as incorrect. Such trials are rare for all models except DeepSeek V4.1 Flash, which regularly produces long reasoning traces without ever calling a tool or exhausts its interaction turns (GPT-5.6 Luna

abandons at most one trial per run, and the remaining models none). Table 2 lists the number of abandoned trials for DeepSeek V4.1 Flash in every active run, together with its accuracy when these trials are counted as incorrect and when they are excluded. Excluding them raises its accuracy by up to 17 points but leaves it far below random-K (52.0, 79.7, 90.0 and 92.0% on the official items of P-ARC, ConceptARC, ARC-AGI-1 and ARC-AGI-2). Across all models, excluding abandoned trials reduces the average gap between Active-K and random-K from 12.8 to 11.0 points on the official items and from 13.6 to 11.5 points on the sampled items, and Active-K remains below random-K in all 20 cells.

				Official		Sampled	
Benchmark	Setting	Trials	Abandoned	incl.	excl.	incl.	excl.
P-ARC	Active-K (draw 1)	50	22	16.0	28.6	22.0	39.3
	Active-K (draw 2)	50	10	24.0	30.0	28.0	35.0
	Active-Free	50	13	24.0	32.4	38.0	51.4
ConceptARC	Active-K (draw 1)	160	13	45.0	49.0	67.5	73.5
	Active-K (draw 2)	160	18	50.0	56.3	67.5	76.1
	Active-Free	160	10	59.4	63.3	73.8	78.7
ARC-AGI-1	Active-K (draw 1)	400	56	65.2	75.9	69.5	80.8
	Active-K (draw 2)	400	47	66.5	75.4	68.5	77.6
	Active-Free	400	33	71.2	77.7	71.5	77.9
ARC-AGI-2	Active-K (draw 1)	200	31	59.0	69.8	62.0	73.4
	Active-K (draw 2)	200	27	65.0	75.1	69.5	80.3
	Active-Free	200	19	66.5	73.5	70.5	77.9

Table 2: Abandoned trials of DeepSeek V4.1 Flash in the active settings, and its accuracy (%) when abandoned trials are counted as incorrect (incl.) or excluded from the denominator (excl.).

A.4 Larger Query Budget

To test whether the Active-K budget is simply too tight, we repeat the Active-K setting on P-ARC with two additional queries per task (K+2), so that the agent ends up with two more input-output pairs than the official task and the random-K baseline provide. Table 3 compares the results with Active-K and random-K. The additional queries help Claude Opus 5 considerably (11 points on the official items), but change the accuracy of the other models by at most four points, and all models remain below random-K on both item types. DeepSeek V4.1 Flash abandons 16 of the 50 trials in this setting, which are again counted as incorrect.

	Official			Sampled		
Model	Active-K	K+2	Random-K	Active-K	K+2	Random-K
GPT-6 Astra	72.0	74.0	80.0	83.0	84.0	90.0
Claude Opus 5	47.0	58.0	62.0	55.0	68.0	73.0
Gemini 3.8 Flash	50.0	46.0	55.0	54.0	54.0	69.0
DeepSeek V4.1 Flash*	20.0	24.0	52.0	25.0	36.0	63.0
GPT-5.6 Luna	9.0	8.0	11.0	11.0	18.0	23.0

Table 3: Accuracy (%) on P-ARC with the regular Active-K budget, two additional queries (K+2), and the random-K baseline. Active-K and random-K are averaged over two draws, K+2 is a single run. *Abandoned trials are counted as incorrect.

A.5 Slippage

Many ARC-AGI-1 tasks admit several verifier implementations that agree on the official examples but diverge on a broader input distribution. We use this to construct a setting, which we call slippage, in which the initial sample is consistent with a narrower rule than the one that actually answers the queries. For each task, we pair the broad RE-ARC verifier with a narrower verifier from one of the code-golf collections, which reproduces the official and ARC-GEN examples but fails on the

majority of RE-ARC samples, and draw the initial sample from the ARC-GEN generator such that it is consistent with both. All queries are then answered by the broad verifier, and the single test item is drawn from the RE-ARC generator. Out of 249 tasks for which such a pair exists, we use the 230 tasks whose broad verifier is also the one used in our main experiments. The setting otherwise corresponds to Active-Free, including the prompt. Table 4 compares the results with the sampled items of the regular Active-Free setting on the same tasks. Accuracy drops for every model, by 10 to 22 points, while the number of queries remains essentially unchanged, suggesting that models rarely probe beyond the input distribution suggested by the initial sample. Note that the test items of the two settings come from different generators, so part of the difference may also reflect a difference in difficulty.

	Accuracy (%)		Queries per task	
Model	Slippage	Active-Free	Slippage	Active-Free
GPT-6 Astra	83.5	93.5	1.25	1.17
Claude Opus 5	70.0	86.5	1.29	1.10
Gemini 3.8 Flash	75.2	89.6	7.91	7.99
DeepSeek V4.1 Flash*	52.6	70.9	5.00	4.34
GPT-5.6 Luna	25.7	47.4	1.10	1.22

Table 4: Accuracy and mean number of queries on 230 ARC-AGI-1 tasks when the initial sample is consistent with a narrower rule (slippage), compared to the sampled items of the regular Active-Free setting on the same tasks. *DeepSeek V4.1 Flash abandons 18 trials in the slippage setting, which are counted as incorrect (57.1% when excluded).

A.6 Program Induction

In this variant of the Active-Free setting, the agent does not predict an output grid at the end of the interaction, but submits a Python program implementing the rule. The exploration phase and the prompt are otherwise unchanged, except that the agent is told that its program will be run on held-out examples of the rule and only counts as correct if it reproduces every one of them. We evaluate each program on the official training and test pairs, a frozen set of generator samples, and up to 50 fresh generator samples, with a timeout of two seconds per call. This criterion is considerably stricter than predicting a single output, so the accuracies in Table 5 are not directly comparable to the main results. More interesting is the change in behavior: GPT-6 Astra and Claude Opus 5 submit about twice as many queries as in the regular Active-Free setting and skip querying in far fewer tasks, while Gemini 3.8 Flash, which already queries extensively, changes comparatively little.

			Queries per task		No queries (%)	
Benchmark	Model	Program correct (%)	Program	Active-Free	Program	Active-Free
P-ARC	GPT-6 Astra	28.0	2.52	1.18	14	30
	Claude Opus 5	16.0	2.36	1.48	22	44
	Gemini 3.8 Flash	20.0	4.26	3.90	50	36
ConceptARC	GPT-6 Astra	35.6	2.49	1.16	7	39
	Claude Opus 5	23.1	1.96	0.95	12	43
	Gemini 3.8 Flash	36.2	7.19	5.71	17	8
ARC-AGI-1	GPT-6 Astra	75.8	2.47	1.00	17	44
	Claude Opus 5	58.2	2.23	0.93	22	53
	Gemini 3.8 Flash	70.0	9.04	7.08	16	12
ARC-AGI-2	GPT-6 Astra	79.0	2.40	1.11	14	42
	Claude Opus 5	61.0	2.38	1.01	26	52
	Gemini 3.8 Flash	72.5	7.80	6.91	18	15

Table 5: Program induction variant of the Active-Free setting: share of tasks for which the submitted program reproduces all held-out examples, mean number of queries per task, and share of tasks without any query, compared to the regular Active-Free setting.

A.7 Higher Reasoning Effort

All main experiments use low reasoning effort. To test whether more deliberation changes the picture, we repeat the Active-Free setting on P-ARC with GPT-6 Astra at high reasoning effort. Table 6 compares both runs. With high effort, the model queries somewhat more often (1.8 compared to 1.2 queries per task) and skips querying in slightly fewer tasks, and its queries are somewhat larger, although still smaller than both random samples (198 cells) and the authored training inputs (147 cells) on P-ARC. Otherwise, neither accuracy nor the character of the queries changes substantially: the queries are about as diverse as before and, if anything, slightly more often near-copies of the initial sample, despite roughly four times as many reasoning tokens.

	Low effort	High effort
Official accuracy (%)	68.0	66.0
Sampled accuracy (%)	78.0	82.0
Queries per task	1.18	1.84
Tasks without queries (%)	30	22
Mean query area (cells)	72	115
Pairwise distance	0.78	0.76
Near-copies of the initial sample (%)	12	19
Median reasoning tokens per trial	434	1,626

Table 6: GPT-6 Astra in the Active-Free setting on P-ARC with low (main experiments) and high reasoning effort (single run each). Query statistics are computed over the accepted queries of tasks with at least one query, as in Table 7.

A.8 Evidence Size, Diversity and Query Counts

Table 7 breaks down the comparison of Section 4.2 by model and benchmark, and Figure 6 shows the distribution of the number of queries per task in the Active-Free setting. Model-constructed queries are smaller than both the random samples and the authored training inputs in almost every cell, whereas lower diversity is concentrated on P-ARC and ConceptARC. In the Active-Free setting, GPT-6 Astra, Claude Opus 5 and GPT-5.6 Luna do not query at all in 42 to 50% of all tasks and rarely submit more than two queries, while Gemini 3.8 Flash submits five or more queries in about half of all tasks.

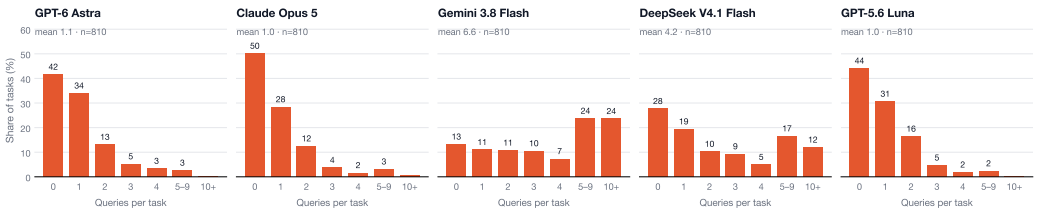

Figure 6: Distribution of the number of accepted queries per task in the Active-Free setting, pooled over all four benchmarks (810 tasks per model).

A.9 Rejected Queries

Queries on which the verifier fails, or which are not valid grids, are rejected and do not count towards the query budget. Table 8 reports the share of rejected queries, the share of trials with at least one rejected query, and the accuracy on the official items of trials with and without a rejected query. Overall, 21.2% of all submitted queries are rejected in the Active-K setting and 14.5% in the Active-Free setting, with considerably higher rates on P-ARC (26 to 66%) and ConceptARC (18 to 44%) than on ARC-AGI-1 and ARC-AGI-2 (5 to 18%). Trials with at least one rejected query are less accurate than trials without one in 16 of 20 cells in the Active-K setting (by 4.7 points on average) and in 15 of 20 cells in the Active-Free setting (by 8.8 points). Since more difficult tasks may lead to both more rejected queries and more errors, this comparison is correlational.

			Mean input area			Pairwise distance			Near-copies (%)	
Benchmark	Model	Trials	Queries	Random	Authored	Queries	Random	Authored	Queries	Random
P-ARC	GPT-6 Astra	100	90	198	147	0.79	0.96	0.89	9	0
	Claude Opus 5	100	134	190	147	0.51	0.95	0.89	25	2
	Gemini 3.8 Flash	100	138	207	147	0.44	0.95	0.89	38	2
	DeepSeek V4.1 Flash	84	150	189	140	0.40	0.95	0.89	33	1
	GPT-5.6 Luna	98	130	193	145	0.43	0.93	0.89	46	3
ConceptARC	GPT-6 Astra	320	63	178	114	0.88	0.92	0.79	2	1
	Claude Opus 5	320	89	172	114	0.70	0.90	0.79	12	1
	Gemini 3.8 Flash	320	70	166	114	0.69	0.91	0.79	14	1
	DeepSeek V4.1 Flash	306	111	175	113	0.55	0.91	0.79	19	1
	GPT-5.6 Luna	319	103	172	114	0.69	0.90	0.79	20	1
ARC-AGI-1	GPT-6 Astra	800	50	171	144	0.88	0.68	0.66	3	6
	Claude Opus 5	800	63	171	144	0.73	0.68	0.66	6	6
	Gemini 3.8 Flash	800	34	171	144	0.82	0.68	0.66	5	6
	DeepSeek V4.1 Flash	782	83	169	142	0.59	0.68	0.66	15	6
	GPT-5.6 Luna	799	63	171	144	0.77	0.68	0.66	10	6
ARC-AGI-2	GPT-6 Astra	400	53	139	136	0.89	0.67	0.74	3	9
	Claude Opus 5	400	67	164	136	0.76	0.75	0.74	7	7
	Gemini 3.8 Flash	400	41	164	136	0.81	0.75	0.74	7	7
	DeepSeek V4.1 Flash	379	81	163	135	0.61	0.75	0.74	16	7
	GPT-5.6 Luna	399	66	163	136	0.78	0.74	0.74	9	7

Table 7: Size and diversity of the evidence in the Active-K setting (both draws pooled) compared to random-K samples (one draw) and the human-authored training inputs, on tasks for which all three are available (Trials counts Active-K trials over both draws). Area is the mean number of cells of the inputs other than the initial sample (for authored evidence, of all training inputs). Pairwise distance is averaged over all pairs of inputs of a trial including the initial sample, and is 1 for inputs of different shape and the fraction of differing cells otherwise. Near-copies are inputs of the same shape as the initial sample with at most 10% of cells changed.

		Active-K				Active-Free			
		Rejected	Trials with	Accuracy (%)		Rejected	Trials with	Accuracy (%)	
Benchmark	Model	queries (%)	rejection (%)	with	without	queries (%)	rejection (%)	with	without
P-ARC	GPT-6 Astra	26.8	39	66.7	75.4	29.8	28	57.1	72.2
	Claude Opus 5	44.7	54	44.4	50.0	33.9	34	47.1	60.6
	Gemini 3.8 Flash	44.8	69	46.4	58.1	26.1	64	56.2	66.7
	DeepSeek V4.1 Flash*	66.0	45	28.9	12.7	31.2	46	13.0	33.3
	GPT-5.6 Luna	51.8	66	10.6	5.9	52.6	52	11.5	8.3
ConceptARC	GPT-6 Astra	21.3	40	80.6	85.3	18.1	20	68.8	82.0
	Claude Opus 5	40.5	48	67.8	75.6	24.4	17	51.9	72.9
	Gemini 3.8 Flash	34.2	59	69.5	73.1	28.4	84	82.8	65.4
	DeepSeek V4.1 Flash*	43.5	51	46.9	48.1	36.3	52	61.4	57.1
	GPT-5.6 Luna	42.1	62	24.9	39.8	39.1	32	26.9	37.0
ARC-AGI-1	GPT-6 Astra	6.5	10	84.8	93.8	8.3	5	75.0	93.2
	Claude Opus 5	11.9	12	76.0	86.1	9.3	4	72.2	83.8
	Gemini 3.8 Flash	11.0	17	82.1	83.0	6.6	25	88.0	87.3
	DeepSeek V4.1 Flash*	17.7	15	49.2	68.9	6.1	11	64.4	72.1
	GPT-5.6 Luna	17.4	21	32.2	48.3	12.8	11	37.2	45.7
ARC-AGI-2	GPT-6 Astra	8.1	13	92.3	88.5	5.9	6	63.6	93.1
	Claude Opus 5	10.4	13	72.5	79.9	5.2	4	57.1	73.1
	Gemini 3.8 Flash	12.6	21	83.5	74.9	5.8	25	84.0	83.3
	DeepSeek V4.1 Flash*	9.4	14	59.6	62.4	10.3	9	61.1	67.0
	GPT-5.6 Luna	15.8	22	25.6	28.7	11.9	12	29.2	29.5

Table 8: Rejected queries in the active settings: share of all submitted queries that are rejected, share of trials with at least one rejected query, and accuracy (%) on the official items of trials with and without a rejected query. Active-K is pooled over both draws, Active-Free is a single run. *Abandoned trials are counted as incorrect.

A.10 Evidence Replay

To separate the quality of the evidence gathered in the Active-K setting from the inference performed on it, we present the evidence of each model's Active-K trials on P-ARC to GPT-6 Astra as fixed demonstrations, analogous to the static setting, and let it answer the same test items. We compare two conditions: the initial sample alone, and the initial sample together with all queries of the

respective trial. Table 9 shows that adding the queries changes accuracy by at most three points on the official items, for every source model including GPT-6 Astra itself. For reference, GPT-6 Astra reaches 80.0% (official) and 90.0% (sampled) on P-ARC with the same number of random generator samples.

	Official		Sampled	
Evidence from	Initial sample	+ queries	Initial sample	+ queries
GPT-6 Astra	64.0	66.0	71.0	77.0
Claude Opus 5	62.0	63.0	71.0	75.0
Gemini 3.8 Flash	60.0	63.0	68.0	66.0
GPT-5.6 Luna	67.0	66.0	72.0	70.0

Table 9: Accuracy (%) of GPT-6 Astra on P-ARC when given the evidence of another model's Active-K trial as fixed demonstrations, either the initial sample alone or together with all accepted queries. Averaged over the two Active-K draws of each source model.

A.11 Noisy Observations

In this variant of the Active-Free setting on ARC-AGI-1, each query output is replaced with probability 0.25 by a corrupted output, generated either by a mutated version of the verifier, by permuting colors of the correct output, or by taking the output of a different instance of the same task, and discarded if it happens to match the correct output. The agent is not told which outputs are affected. Table 10 summarizes the results. Neither model loses significantly in accuracy, and both re-submit the same input far more often after a corrupted output than after a correct one, suggesting that they notice implausible observations and verify them.

	Accuracy (%)				Re-queried (%)	
Setting	Official	Sampled	Queries	Corrupted (%)	after corrupted	after correct
GPT-6 Astra, clean	92.2	94.0	1.00	–	–	0.0
GPT-6 Astra, noisy	93.8	94.0	1.61	28	32.3	2.4
Gemini 3.8 Flash, clean	87.5	90.2	7.08	–	–	0.4
Gemini 3.8 Flash, noisy	84.8	86.0	6.85	26	41.8	8.8

Table 10: Active-Free on ARC-AGI-1 without (clean) and with (noisy) corruption of 25% of all query outputs: accuracy, mean number of queries per task, share of corrupted query outputs, and share of outputs after which the same input is submitted again. By corruption type, GPT-6 Astra re-queries 26% of outputs from mutated verifiers, 26% of color permutations and 40% of outputs from a different instance; Gemini 3.8 Flash 43%, 39% and 43%. Two Gemini trials in the noisy setting are abandoned and counted as incorrect.

A.12 Teacher Demonstrations

Figure 7 contrasts the demonstrations of GPT-6 Astra and GPT-5.6 Luna for ARC-AGI-1 task 27a28665, on which all five students solve the test items when taught by Astra and none when taught by Luna, although Luna itself solves them in its exam.

A.13 Teaching and the Teacher Exam

Table 11 splits the teaching results by whether the teacher solved the official test items of a task in its own exam before teaching. Since raw student accuracy is naturally higher on tasks that are easy enough for the teacher to solve, we focus on the difference to the students' static accuracy on the same tasks. For Gemini 3.8 Flash, DeepSeek V4.1 Flash and GPT-5.6 Luna, this difference is nearly identical whether or not the teacher passed its exam (averaged over benchmarks, -8.7 compared to -8.2, -10.0 compared to -9.7, and -15.3 compared to -18.6 points), i.e., their demonstrations are similarly uninformative even on tasks they can solve themselves. Claude Opus 5 teaches considerably worse on the tasks it fails (-18.5 compared to -7.3 points), although these

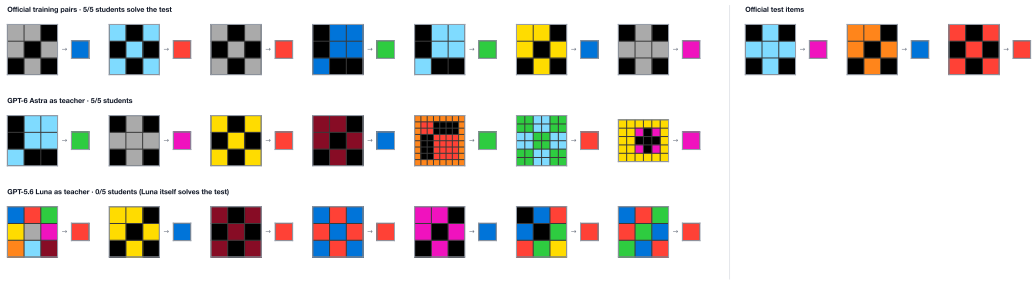

Figure 7: Official training pairs and the demonstrations constructed by GPT-6 Astra and GPT-5.6 Luna for ARC-AGI-1 task 27a28665, in which each shape pattern maps to a specific color. Astra's demonstrations cover all four patterns (partly embedded in larger grids), whereas Luna's cover only two of them and add several full grids, so the plus-shaped pattern of the first test item never appears.

are comparatively few, and GPT-6 Astra fails its exam on only 8 of 810 tasks, so its failed cases are not informative.

		Teacher passed exam			Teacher failed exam		
Benchmark	Teacher	Tasks	Student acc. (%)	Δ static	Tasks	Student acc. (%)	Δ static
P-ARC	GPT-6 Astra	50	57.8	+0.4	0	–	–
	Claude Opus 5	46	44.3	−11.3	4	35.0	−42.5
	Gemini 3.8 Flash	15	51.3	−10.7	35	42.3	−13.1
	DeepSeek V4.1 Flash	44	44.3	−15.9	6	30.0	−6.7
	GPT-5.6 Luna	16	33.1	−24.4	34	34.4	−22.9
ConceptARC	GPT-6 Astra	154	80.5	−2.6	6	76.7	−1.7
	Claude Opus 5	130	77.1	−8.1	30	59.3	−14.0
	Gemini 3.8 Flash	127	73.2	−12.0	33	66.7	−7.6
	DeepSeek V4.1 Flash	145	73.9	−9.7	15	60.0	−16.7
	GPT-5.6 Luna	81	71.6	−15.9	79	62.8	−15.4
ARC-AGI-1	GPT-6 Astra	399	87.7	+0.4	1	0.0	−80.0
	Claude Opus 5	377	82.3	−6.0	23	53.9	−17.4
	Gemini 3.8 Flash	274	85.1	−7.6	126	63.8	−11.9
	DeepSeek V4.1 Flash	372	78.4	−9.7	28	65.0	−12.1
	GPT-5.6 Luna	197	81.8	−11.3	203	58.3	−23.4
ARC-AGI-2	GPT-6 Astra	199	81.5	−2.2	1	60.0	−20.0
	Claude Opus 5	188	80.3	−3.9	12	75.0	+0.0
	Gemini 3.8 Flash	141	85.0	−4.4	59	69.8	−0.3
	DeepSeek V4.1 Flash	182	79.8	−4.8	18	71.1	−3.3
	GPT-5.6 Luna	103	79.8	−9.7	97	65.2	−12.4

Table 11: Teaching results split by whether the teacher solved the official test items of a task in its exam: number of tasks, student accuracy on the official items averaged over all five students, and the difference to the students' static accuracy on the same tasks (points).

A.14 VARC AS STUDENT

Table 12 lists the full results of the VARC student on all 400 ARC-AGI-1 tasks, trained from scratch on each task with either the official training pairs or the budgeted demonstrations of one of the five teachers. A task counts as solved if all of its official test items (or the sampled item) are solved. Differences to the official demonstrations are paired by task, with 95% bootstrap confidence intervals over tasks.

A.15 EXAMPLE TRAJECTORIES

Figures 8–12 show a selection of trajectories from the active settings that illustrate the behaviors discussed in the main text. Each figure shows the initial sample, all accepted queries in the order in which they were submitted (rejected queries are omitted), and the first official test item together with

Demonstrations	Official items			Sampled items		
	Pass@1	Pass@2	Δ Pass@2	Pass@1	Pass@2	Δ Pass@2
Official	35.8	39.5	–	37.5	41.2	–
GPT-6 Astra	26.0	31.0	−8.5 [-13.2, -3.8]	25.5	31.2	−10.0 [-14.8, -5.5]
Claude Opus 5	25.5	29.2	−10.2 [-15.0, -5.8]	22.0	26.2	−15.0 [-19.5, -10.5]
Gemini 3.8 Flash	15.5	19.2	−20.2 [-24.8, -15.5]	16.5	19.0	−22.2 [-27.0, -17.5]
DeepSeek V4.1 Flash	14.0	16.2	−23.2 [-28.0, -18.8]	17.8	22.2	−19.0 [-24.2, -14.0]
GPT-5.6 Luna	13.2	15.8	−23.8 [-28.2, -19.2]	16.5	18.5	−22.8 [-27.8, -17.8]

Table 12: Share of ARC-AGI-1 tasks (%) solved by VARC trained from scratch on the official training pairs or on the demonstrations of each teacher, and the paired difference to the official demonstrations (pass@2) with 95% bootstrap confidence intervals over tasks.

the model's answer and the correct output. The first two trajectories are taken from the Active-K setting and contrast a query strategy that confirms an incorrect hypothesis with one that systematically covers the relevant factors of the task. The third one shows queries that are valid but uninformative, and the last two contrast extensive and absent experimentation in the Active-Free setting.

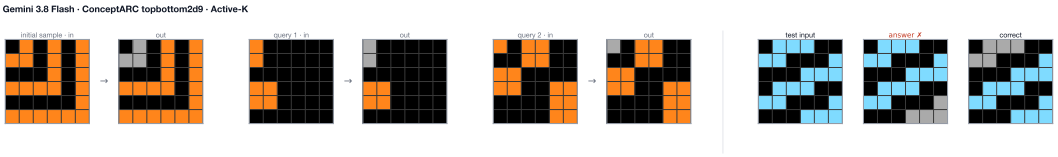

Figure 8: Gemini 3.8 Flash on ConceptARC task topbottom2d9 (Active-K). The rule recolors the topmost shape in grey. In the initial sample, the topmost shape happens to also be the smallest one, and both queries preserve this coincidence, so they are consistent with both hypotheses. The model settles on the incorrect one and recolors the smallest shape of the test input instead of the topmost one, failing all three official test items.

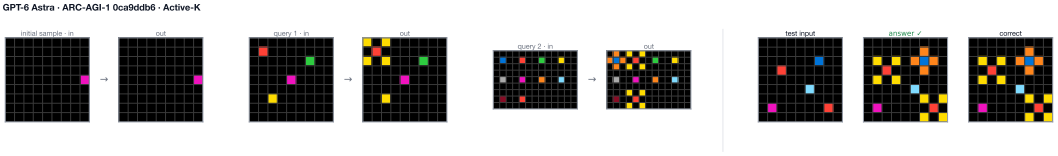

Figure 9: GPT-6 Astra on ARC-AGI-1 task 0ca9ddb6 (Active-K). The initial sample contains only a single magenta cell, which the transformation leaves unchanged, and therefore reveals almost nothing about the rule. The model uses its two queries to place cells of every color on an otherwise empty grid, which reveals that only red (diagonal yellow neighbors) and blue (orthogonal orange neighbors) cells have an effect, and solves the test item.

A.16 Prompts

Below we reproduce the system prompts of the active settings and the teaching setting, instantiated for a task with three official training pairs. In the active settings, the prompt is followed by a user message containing the initial sample as JSON; in the teaching setting, by a message containing the natural-language rule, the verifier program and the teacher's example pair. The exam message is followed by the test input as JSON. Grids are always exchanged as JSON lists of integers.

Active-K

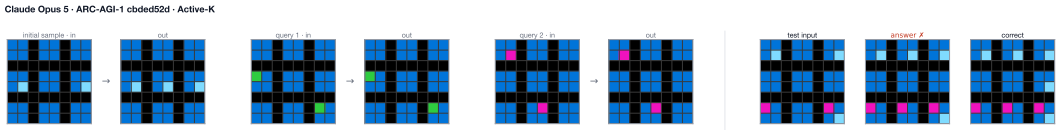

Figure 10: Claude Opus 5 on ARC-AGI-1 task cbded52d (Active-K). Two cells of the same color that are aligned across the block grid fill the corresponding cell of the block between them. The initial sample shows a horizontal alignment, and both queries place colored cells that are not aligned with any other cell of the same color, so neither changes anything. The model never tests a vertical alignment and misses the vertical fill in the test item, while solving the horizontal ones.

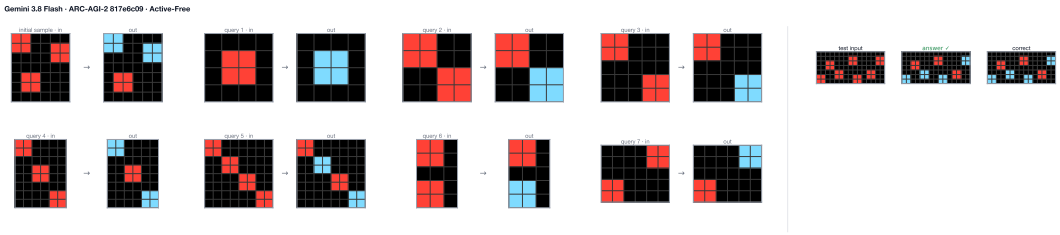

Figure 11: Gemini 3.8 Flash on ARC-AGI-2 task 817e6c09 (Active-Free). Starting from the initial sample, the model submits seven queries that vary the number, position and arrangement of the red squares, isolating which of them are recolored, and solves the test item. In the Active-K setting, with a budget of four queries, it fails the official test item in both draws.

```
In the following, you are given a single example consisting of an
input-output grid pair.  There is an abstract rule that describes the
transformation from the input grid to the output grid.  Your task is to
determine this abstract rule, over a series of steps.

You have a fixed budget of exactly 2 queries:  each one asks for
another input-output grid pair that follows the same rule, for an input
grid you choose.  You will take the test after you have used the
budget, not before, and you will then be given new input grid(s) to
apply the rule to.

- Use the submit_query tool to ask for an input grid.  After the 2
queries are used, call request_test for the final test; it is refused
before then.

- The budget is fixed, so queries cost you nothing and there is no
credit for using fewer.  You are scored only on whether your test
output grid is correct, so spend every query on whatever would tell you
most about the rule.
```

Active-Free

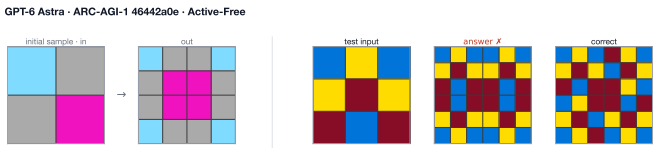

Figure 12: GPT-6 Astra on ARC-AGI-1 task 46442a0e (Active-Free). The output tiles four rotated copies of the input. Because the initial sample is symmetric, rotating and mirroring it produce the same output. The model requests the test without a single query and mirrors the (asymmetric) test input instead of rotating it, whereas one query with an asymmetric input would have distinguished the two hypotheses.

```
In the following, you are given a single example consisting of an
input-output grid pair.  There is an abstract rule that describes the
transformation from the input grid to the output grid.  Your task is to
determine this abstract rule, over a series of steps.

At each step, you may either (1) make a query:  i.e., ask for another
example of an input-output grid pair that follows the same rule, or,
(2) if you are confident you know the rule, you may ask for a test, in
which you will be given a new input grid, and you will need to apply
the rule to generate the correct output grid.

- You can use the submit_query tool to ask for an input grid.  Once you
are confident you know the rule, you can use the request_test tool to
request the final test.

- You are scored on whether you generate a correct output grid when you
are given a test.  The number of queries you submit will also be
recorded.
```

Teacher (budgeted)

```
You are dealing with grid transformation tasks:  an input grid is
mapped to an output grid according to an underlying rule.  Grids are
rectangular matrices with integer colors 0-9.

You are given the rule in words, the verifier program that implements
it, and one input-output example.  Your job is to assemble exactly 3
demonstration pairs of this rule.

Those pairs are then given to a separate solver that has never seen
this task.  It sees your 3 pairs and nothing else:  not the rule, not
the verifier program, not the example you were given, and nothing you
write in prose.  From them alone it must infer the rule and apply it to
held-out inputs of this task.  Whether it answers those correctly is
what is scored.

Use show_transformed_input to add a pair:  you author the input and the
environment computes the output.  An input the verifier cannot evaluate
is refused and no pair is added; you may submit a different one.
```

Teacher exam

```
Before you teach:  apply the rule to this input yourself and submit the
output with submit_prediction.  This checks that you have the rule; you
will not keep this exchange.
```