LLM-AutoSciLab: Closed-Loop Scientific Discovery via Active Experimentation with LLMs

Sanchit Kabra[1*], **Nikhil Abhyankar**[1*], **Saaketh Desai**[2], **Prasad P. Iyer**[2], **Chandan K. Reddy**[1]

[1]Virginia Tech [2]Sandia National Laboratories

Abstract

Scientific discovery is a closed-loop process where hypotheses guide data acquisition and observations refine the hypothesis space. Yet most approaches reduce discovery to supervised learning over fixed datasets, where limited observations can support multiple plausible mechanisms that fit locally but fail to generalize. Thus, the key challenge is selecting informative observations to resolve uncertainty, shifting the focus from static inference to adaptive data acquisition. To address this, we propose **LLM-AutoSciLab**, a closed-loop framework that couples hypothesis generation with hypothesis-conditioned experiment selection and mechanism refinement. Rather than fitting models to passively collected data, LLM-AutoSciLab iteratively proposes plausible hypotheses, selects informative experiments to distinguish or refine them, and updates its state using the resulting evidence. To evaluate dynamic, closed-loop scientific discovery with active data acquisition beyond existing known-variable equation recovery, we introduce **ActiveSciBench**, comprising two datasets: (i) ActiveSciBench-Chem (57 enzyme-kinetics tasks) and (ii) ActiveSciBench-GRN (45 gene-regulatory-network tasks), that model discovery as a budget-constrained process requiring adaptive experiment design, variable selection, and recovery of true mechanisms. Across NewtonBench, ActiveSciBench-Chem, and ActiveSciBench-GRN, LLM-AutoSciLab outperforms prior methods, achieving 67.6% and 35.1% symbolic accuracy on NewtonBench and ActiveSciBench-Chem, respectively, and 31.1% exact graph recovery on ActiveSciBench-GRN. Moreover, comparison methods require 1.3–7.4$\times$ as many oracle queries to match LLM-AutoSciLab's fixed-budget recovery.[1]

1 Introduction

Discovering governing mechanisms underlying physical systems remains a central challenge in science [47, 37]. Recent advances in large language models (LLMs) have enabled scientific agents that leverage pretrained knowledge, reasoning, and tool use to generate hypotheses, search over candidate models, and interpret observed data [48, 5, 40]. However, *existing methods treat scientific discovery as static, supervised inference on fixed datasets* [14, 44], searching for hypotheses that best fit a limited set of observations. This static formulation creates a fundamental *identifiability bottleneck*: multiple hypotheses can fit the observed data equally well yet diverge under unseen conditions, preventing reliable recovery of the true underlying law [27]. Scientific discovery is instead inherently a closed loop, with hypotheses guiding experiments and observations refining subsequent hypotheses [9]. Crucially, informative experiments can deliberately induce conditions under which competing explanations make divergent predictions, exposing mechanistic distinctions that limited or biased observations cannot resolve [8, 36].

[*]Equal contribution. Correspondence: sanchit23@vt.edu, nikhilsa@vt.edu.

[1]Code: https://github.com/scientific-discovery/LLM-AutoSciLab

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

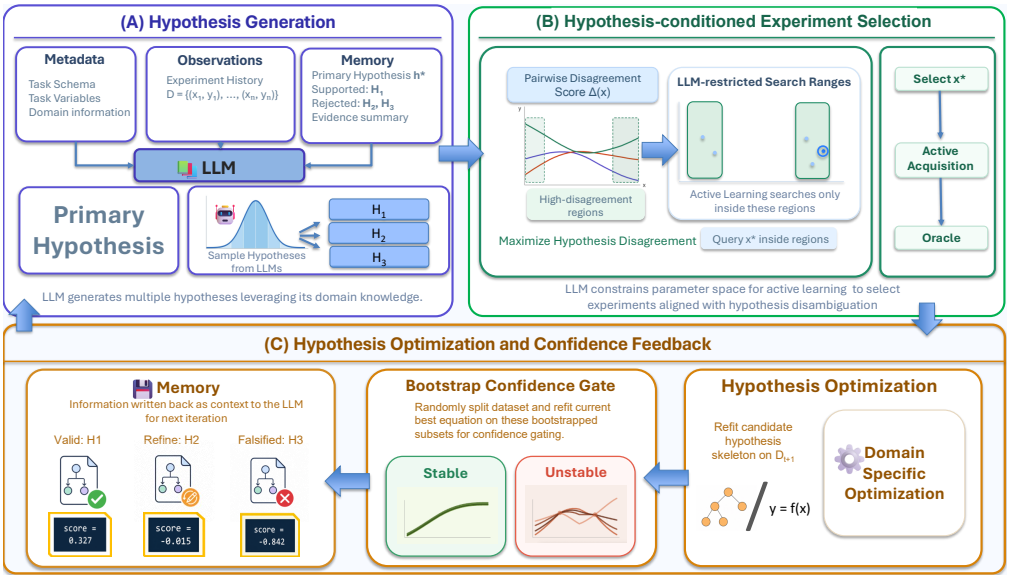


Figure 1: **Overview of LLM-AutoSciLab.** (A) An LLM generates candidate hypotheses from observations and memory. (B) Experiments are actively selected in regions of maximal disagreement with the hypothesis. (C) Candidates are iteratively refined via domain-specific optimization (e.g., parameter fitting and constraint enforcement), with confidence-based feedback guiding updates.

Recent self-driving laboratories increasingly automate parts of this loop, including experimental optimization, intervention selection, hypothesis generation, and laboratory-in-the-loop refinement [32, 29, 16]. However, they still require substantial human effort to formulate and refine hypotheses. Classical active learning selects observations to reduce predictive uncertainty, while Bayesian experimental design can discriminate among competing models through expected utility, but both typically assume that the relevant model or candidate space is known and specified in advance. This creates a central bottleneck for mechanism discovery, since the hypothesis space is itself unknown and must be expanded or revised as evidence accumulates, rather than merely selected within a fixed set of candidates. To address this gap, we propose **LLM-AutoSciLab**, *a closed-loop framework that models scientific discovery as active hypothesis-conditioned experiment design rather than passive regression over fixed datasets* (Table 1). Unlike classical approaches which optimize acquisition within a predefined hypothesis space, LLM-AutoSciLab *jointly evolves the mechanism space and selects experiments that refine it through feedback.* At iteration t, LLM-AutoSciLab constructs a diverse set of structured mechanisms from accumulated evidence and identifies experimental regions where their predictions diverge. LLM-AutoSciLab selects new experiments using a *hypothesis-conditioned acquisition objective* that directly targets these disagreements, prioritizing observations that are most informative for distinguishing among competing laws (Figure 1). Each new observation is then used to eliminate, refine, or restructure the hypothesis set, which in turn determines the next experiment. This creates a tight feedback loop between *what mechanisms are currently plausible* and *what evidence should be collected next*, allowing each query to be spent on resolving the ambiguities that prevent identification of the underlying law. By explicitly optimizing experiments for mechanism disambiguation, LLM-AutoSciLab enables sample-efficient law recovery under constrained experimental budgets.

Existing benchmarks [52, 11, 50] enable sequential experimentation but remain largely equation-centric and do not capture the full challenge of *active mechanism discovery* (see Table 2). In particular, they alter canonical physical laws, producing governing equations that depart from real-world physics and may be physically implausible by design. Scientific discovery, however, extends beyond recovering equations and requires an agent to determine what evidence to acquire, identify which variables or interactions are mechanistically relevant, and recover the underlying scientific structure under a finite experimental budget. **ActiveSciBench** targets this broader notion of discovery through scientifically grounded systems. It provides budget-limited oracle access while withholding the relevant variables, requiring agents to jointly determine *what to measure, what matters, and what mechanism explains the observations*. ActiveSciBench-Chem requires recovery of enzyme-kinetic mechanisms from actively selected reaction conditions in the presence of distractor variables, whereas ActiveSciBench-GRN requires intervention-driven recovery of directed, signed regulatory structure

from perturbation responses. Together, they *extend active scientific-discovery evaluation beyond symbolic equation fitting to heterogeneous mechanism discovery under constrained budgets*. We evaluate LLM-AutoSciLab using `GPT-4o-mini` and `Qwen-3-4B/14B/32B`, demonstrating that it discovers governing mechanisms faster across settings. Our main contributions are:

- We introduce LLM-AutoSciLab, a *closed-loop scientific discovery framework* coupling LLM-guided hypothesis generation and refinement with hypothesis-conditioned experiment design.
- We introduce ActiveSciBench, *a benchmark suite for active mechanism discovery in scientifically grounded systems*, with budget-limited oracle access and hidden variable selection.
- We propose a *hypothesis-conditioned acquisition strategy* that selects experiments maximizing disagreement among competing hypotheses, improving sample efficiency under fixed budgets.
- We show that LLM-AutoSciLab outperforms prior methods across benchmarks, achieving up to 67.6% symbolic accuracy and 31.1% exact graph recovery; comparison methods require 1.3–7.4$\times$ as many oracle queries to match its fixed-budget recovery. Ablations confirm the importance of each component.

Table 1: Comparison of scientific discovery frameworks across key design dimensions.

Method / Framework	Experimental Design	Feedback-driven Refinement	Closed-Loop Discovery	Representation Agnostic	Primary Output
Passive law discovery [47, 14]	✗	✗	✗	✗	Equation
LLM-guided equation discovery [44, 7]	✗	✓	✗	✗	Equation
General experiment design [36]	✓	✓	✗	✗	Query policy
Domain-specific adaptive discovery [32, 6]	✓	✓	✗	✗	Optimized condition
Self-driving laboratories [3, 33]	✓	✓	✓	✗	Domain objective
Interpretable SDLs [16, 31]	✓	✓	✓	✗	Equation / model
Biological network inference [38, 43]	✗	✓	✗	✗	Causal graph
LLM-AutoSciLab (Ours)	✓	✓	✓	✓	**Open-ended mechanism**

2 Related Work

LLMs for Scientific Discovery. LLMs have shown strong potential for scientific discovery through hypothesis generation and refinement [54, 26, 20], data-driven analysis [34, 40, 4], and equation discovery [44, 21, 7, 51]. LLM-guided search has also been applied across chemistry [49], materials discovery [1, 17], and program synthesis [41]. These systems demonstrate the ability of LLMs to propose and refine scientific explanations across diverse representations. However, the scientific evidence available to the agent is often fixed or externally provided, leaving experiment selection outside the loop. LLM-AutoSciLab instead uses explicit mechanism hypotheses not only as objects of reasoning, but also as the basis for deciding what evidence should be acquired next.

Experiment Design for Scientific Discovery. Experimental design formalizes discovery as selecting measurements or interventions that are informative about an unknown system under limited budgets [36], with applications across materials and process optimization [32, 29], drug and molecular discovery [6, 30], genomics and perturbation screening [24, 39], and applied physics [35]. Self-driving laboratories extend this paradigm to physical closed-loop platforms by coupling adaptive decision-making with automated experimentation [3, 33]. Recent work increasingly automates experiment selection [16], and integrates LLMs for biological perturbation design [42] along with iterative hypothesis refinement from laboratory feedback [19]. [13] uses LLM-derived beliefs within sequential Bayesian experimental design to actively gather informative evidence, whereas [27, 2], explore this setting for differential equations. LLM-AutoSciLab instead operationalizes discovery through an explicit population of competing mechanisms whose predictive disagreement directly determines where to experiment next, coupling hypothesis generation, mechanism discrimination, and revision within a single loop.

Benchmarks for Scientific Discovery. Existing discovery benchmarks, including classical [47, 14], and recent ones targeting memorization resistance and distribution shift [45, 28], largely evaluate offline recovery from fixed observations with supplied variables and equation-based targets. Recent interactive benchmarks move toward sequential discovery: BoxingGym couples model proposal with experimentation, while PhysGym, NewtonBench, and DiscoverPhysics evaluate discovery through interactive physics environments [18, 11, 52, 50]. These datasets deliberately alter canonical physical laws to create counterfactual laws, emphasizing recovery of unfamiliar governing laws rather than scientifically grounded mechanisms. In contrast, **ActiveSciBench** evaluates active mechanism discovery in scientifically grounded biochemical and gene-regulatory systems, requiring agents to identify relevant variables, design informative experiments, and recover either functional laws or directed, signed regulatory structure under a fixed budget.

Table 2: Comparison of scientific discovery benchmarks across key properties.

Benchmark / Suite	Active Exploration	Perturbational / Interventional	Physical Grounding	Variable Selection	Memorization Free	Discovery Target
Passive law benchmarks [47, 14]	✗	✗	✓	✗	✗	Closed-form law
LLM equation benchmarks [45]	✗	✗	✗	✗	✓	Out-of-Distribution
Interactive law discovery [52]	✓	✗	✗	✗	✓	Counterfactual law
Dynamical-system benchmarks [46, 15]	✗	✗	✓	✗	✓	Dynamical behavior
Optimization benchmarks [22]	✓	✓	✓	✗	✓	Condition optimization
Causal perturbation benchmarks [12]	✗	✓	✓	✗	✓	Causal structure
GRN inference benchmarks [38, 43]	✗	✓	✓	✗	✓	GRN inference
ActiveSciBench (Ours)	✓	✓	✓	✓	✓	**Active mechanism discovery**

3 LLM-AutoSciLab Method

We instantiate LLM-AutoSciLab as an iterative algorithm over a dynamically maintained hypothesis set. At each iteration, candidate mechanisms are constructed as competing hypotheses that remain consistent with the current evidence, and the next experiment is selected by maximizing their expected disagreement under the available experimental budget. The resulting observation is first evaluated against the candidate predictions, then incorporated by refitting each hypothesis on the augmented dataset, computing its empirical loss, and filtering the hypothesis set to retain mechanisms supported by the new evidence while eliminating those that are falsified or unstable.

3.1 Problem Formulation

We frame scientific discovery as a budgeted active mechanism discovery problem that jointly couples hypothesis generation with adaptive experimental design. Let $\mathcal{M}$ denote a space of candidate mechanisms, where each mechanism $m \in \mathcal{M}$ defines a predictive mapping $f_m : \mathcal{X} \to \mathcal{Y}$. The objective is to recover the unknown ground-truth mechanism $m^\star \in \mathcal{M}$ through a sequence of adaptively chosen experiments. At each round t, the learner selects an experiment $\boldsymbol{x}_{t+1} \in \mathcal{X}$ that yields an observation $y_{t+1} \sim p(\cdot \mid \boldsymbol{x}_{t+1}, m^\star)$, where p is the observation model. Unlike static discovery, where the data are fixed a priori, this task requires active data acquisition. Consequently, after t rounds, the accumulated dataset $\mathcal{D}_t = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^t$ grows iteratively until the total experimental budget B is exhausted. At each step, the process maintains a discovery state $\mathcal{S}_t = (\mathcal{D}_t, \mathcal{E}_t, \mathcal{H}_t)$, where $\mathcal{E}_t$ summarizes prior hypotheses and experimental evidence, and $\mathcal{H}_t = \{m^{(k)}\}_{k=1}^K \subseteq \mathcal{M}$ is a finite approximation to the mechanisms currently compatible with $\mathcal{D}_t$. Rather than committing to a single best model, discovery proceeds by progressively resolving ambiguity among these competing explanations through informative experiments. A policy π maps the current state to the next query, $\boldsymbol{x}_{t+1} = \pi_t(\mathcal{S}_t)$, which produces observations $y_{t+1} \sim p(\cdot \mid \boldsymbol{x}_{t+1}, m^\star)$ and updated states $\mathcal{S}_{t+1}$. The objective is to produce a final estimate $\hat{m}_B$ that minimizes expected mechanism error $\mathbb{E}[\mathcal{L}(\hat{m}_B, m^\star)]$, where $\mathcal{L}$ is a domain-dependent loss, and the expectation is over trajectories induced by π and p, with a fixed $m^\star$.

Algorithm 1 LLM-AutoSciLab

Require: Oracle $\mathcal{O}$, Dataset $\mathcal{D}$, Budget B, State $\mathcal{S}_t$, Search Region $\mathcal{R}$, Memory $\mathcal{E}$, LLM π_θ, Hypothesis Set $\mathcal{H}_t$, Confidence Threshold τ_{conf}, Confidence Score c_t

```
# Initialize data and experience buffer
D_0, c_0, E_0 ← ∅, 0, InitMemory()
for t = 0, ..., B − 1 do
    S_t ← (D_t, E_t, H_t)
    # Propose hypotheses and search regions
    H_t, R_t ← GenHyp(π_θ^large, π_θ^small, S_t)
    # Select acquisition mode
    if c_t < τ_conf then
        mode ← Disambiguate
        Δ_t ← Disagree(H_t, D_t)
    else
        mode ← Refine
        Δ_t ← ∅
    end if
    # Acquire new experiment
    x_{t+1} ← Acquire(D_t, H_t, R_t, mode, Δ_t)
    y_{t+1} ← O(x_{t+1})
    D_{t+1} ← D_t ∪ {(x_{t+1}, y_{t+1})}
    # Refine and update memory
    m̂_{t+1}, c_{t+1} ← RefineHyp(D_{t+1}, H_t)
    m̃_{t+1}, c̃_{t+1} ← ConfGate(m̂_{t+1}, c_{t+1})
    c_{t+1} ← c̃_{t+1}
    E_{t+1} ← UpdateMemory(
      E_t, D_{t+1}, H_t, m̃_{t+1}, c̃_{t+1})
end for
return m̃_B
```

3.2 Hypothesis Generation

Given the accumulated evidence and discovery history, $\texttt{GenHyp}(\cdot)$ constructs a finite set of *rival mechanisms* that remain compatible with the observations while representing distinct scientific

explanations. To reduce path dependence and premature concentration under sparse evidence [10], LLM-AutoSciLab separates *hypothesis exploration* from *hypothesis synthesis*. A smaller LLM $\pi_\theta^{\text{small}}$ repeatedly proposes candidate mechanisms conditioned on $\mathcal{S}_t$; these are grouped into structural families until the empirical distribution stabilizes, yielding $\mathcal{H}_t$. A larger LLM $\pi_\theta^{\text{large}}$ then reasons over $\mathcal{H}_t$ to produce a primary hypothesis, plausible alternatives, and diagnostic regions $\mathcal{R}_t \subseteq \mathcal{X}$ where their predictions are expected to diverge. Thus, $\mathcal{H}_t$ captures mechanisms that are similarly plausible under current evidence but experimentally distinguishable, providing the basis for subsequent acquisition.

3.3 Hypothesis-Conditioned Experiment Selection

The data acquisition policy is governed by the disagreement of the hypotheses $\mathcal{H}_t$ within the LLM-generated search region $\mathcal{R}_t$. Given the confidence score c_t, LLM-AutoSciLab first determines whether the current hypothesis is sufficiently stable to serve as the basis for local refinement. If $c_t < \tau_{\text{conf}}$, LLM-AutoSciLab retains the full hypothesis set $\mathcal{H}_t$ and selects experiments for mechanism disambiguation; otherwise, it treats the current mechanism as stable and shifts acquisition to refinement. In `Disambiguate` mode LLM-AutoSciLab computes $\Delta_t = \texttt{Disagree}(\mathcal{H}_t, \mathcal{D}_t)$, which governs the acquisition strategy and `Acquire(·)` selects $\boldsymbol{x}_{t+1} \in \mathcal{R}_t$ where candidate mechanisms in $\mathcal{H}_t$ make maximally divergent predictions. In `Refine` mode, `Acquire(·)` instead selects experiments that improve the fit or parameterization of the supported mechanism family. Unlike Bayesian experimental design, which typically optimizes information gain under a predefined probabilistic model class, LLM-AutoSciLab constructs and revises explicit mechanism hypotheses and uses their predicted disagreements to guide experiment selection. This makes acquisition depend on the status of the evolving hypothesis space, rather than predictive improvement alone.

3.4 Hypothesis Optimization and Confidence Feedback

After each experiment, the new observations from $\mathcal{O}$ are incorporated into $\mathcal{D}_t$ to produce the new dataset $\mathcal{D}_{t+1}$. As shown in Algorithm 1, `RefineHyp(·)` fits $\mathcal{H}_t$ to the new dataset producing a refined mechanism $\hat{m}_{t+1}$ along with a confidence score c_{t+1}. This step turns each generated hypothesis structure into a data-evaluated mechanism on $\mathcal{D}_{t+1}$. To assess robustness under adaptive data collection, we introduce a confidence gate applied via `ConfGate(·)`. Since $\mathcal{D}_{t+1}$ may be biased toward the currently selected regions, this step performs bootstrap resampling and refits the candidate mechanism across these datasets, measuring agreement across the resulting fits. Consistent hypotheses are assigned higher confidence $\tilde{c}_{t+1}$, while those exhibiting instability in structure or predictions are treated as brittle; the gated score is carried forward as $c_{t+1} \leftarrow \tilde{c}_{t+1}$ and selects the acquisition mode at the next iteration. The confidence-adjusted mechanism $\tilde{m}_{t+1}$ is then written back into memory through `UpdateMemory(·)`, informing subsequent hypothesis generation and acquisition decisions. Appendix E.2 provides a complete algorithmic specification and implementation details.

4 ActiveSciBench: Benchmark for Active Scientific Discovery

Real-world scientific discovery requires not only inferring governing laws from observations, but also choosing experiments that yield the most informative data. Existing benchmarks reduce discovery to passive inference on fixed datasets, bypassing the experimental-design problem that is critical when observations are costly. To address this gap, we introduce **ActiveSciBench**, a two-dataset benchmark suite for active, closed-loop scientific discovery (Figure 2), each based on *physically grounded laws* and a queryable experimental system where the underlying law and parameters are hidden, relevant variables are unknown a priori, and discovery must occur within a fixed experimental budget.

4.1 ActiveSciBench-Chem: Active Enzyme-Kinetic Law Discovery

Task Formulation. Enzyme-kinetic rate laws describe how reaction rates vary as a function of experimental conditions. In ActiveSciBench-Chem (Figure 2 (a)), each task simulates an enzyme-catalyzed reaction governed by a hidden kinetic mechanism and hidden parameters; the learner must recover the symbolic rate law from budget-limited experiments. Each experiment is specified via a shared 7-dimensional interface for substrate, inhibitor, second-substrate, and product concentrations, enzyme loading, temperature, and pH, and returns the observed initial rate r_0 along with auxiliary mass-balance observables. The true rate law, including its functional form and which inputs actually appear, is withheld from the learner throughout. Candidate mechanisms can produce indistinguishable behavior over restricted regions of the design space, and the correct law is often recoverable only through experiments that deliberately isolate individual dependencies.

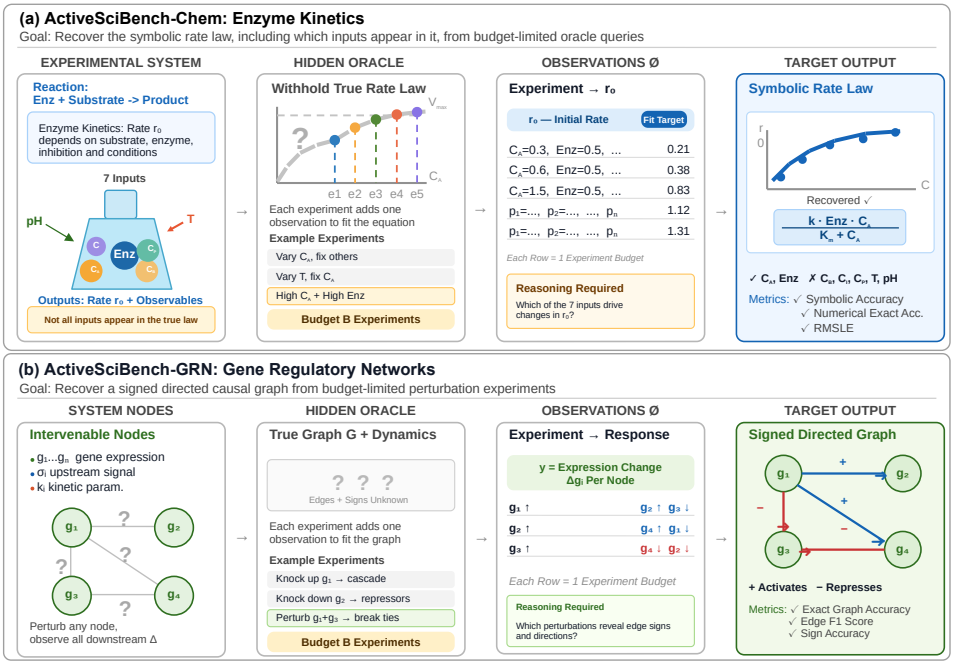


Figure 2: **Overview of ActiveSciBench**: (a) **ActiveSciBench-Chem:** Symbolic enzyme rate law recovery; (b) **ActiveSciBench-GRN:** Signed directed gene regulatory graph inference.

Dataset Construction. The benchmark is drawn from a curated, mechanistically grounded hypothesis space comprising standard kinetic families, structured compositions thereof, and extended mechanisms beyond the standard textbook library, yielding 57 curated tasks in total. We report results across: *Easy* (standard families: Michaelis-Menten, competitive inhibition), *Medium* (structured compositions: Michaelis–Menten with Arrhenius modulation, ping-pong bisubstrate kinetics with noncompetitive inhibition), and *Hard* (extended mechanisms: allosteric activation, cooperative inhibition).

4.2 ActiveSciBench-GRN: Active Causal Graph Discovery

Task Formulation. Gene regulatory networks are signed, directed graphs describing which genes or regulators activate or repress other genes. In ActiveSciBench-GRN (Figure 2 (b)), each task simulates a hidden regulatory system with an unknown graph structure and nonlinear dynamics, aiming to recover the causal graph from within an experimental budget. The system consists of intervenable gene expression nodes, upstream signals, and kinetic parameters. Unlike ActiveSciBench-Chem, ActiveSciBench-GRN operates in a discrete intervention space: each experiment knocks up, knocks down, or perturbs specific nodes, and the learner observes all downstream expression changes. The true graph, including edge presence, direction, and sign (activation vs. repression) along with the governing nonlinear dynamics, is withheld from the learner throughout. Different motifs can produce similar observations under weak interventions, and hidden parameters can make the same motif look qualitatively different across tasks. The learner must therefore jointly identify topology, sign, and effective dynamics from sparse data, requiring informative perturbation choices rather than response-surface fitting. The core challenge is inferring which variables are relevant, but here the discovery target is a structured causal graph rather than a symbolic equation, extending the benchmark suite to broader coverage of scientific discovery problems.

Dataset Construction. The benchmark is curated from a small set of canonical regulatory motifs spanning increasing structural complexity, from feedforward activation to repression, feedback, and switching behavior. It contains 5 motif families, each instantiated at 3 native difficulty levels and 3 topological versions, yielding 45 tasks per random seed with the capacity to generate more. The data split in ActiveSciBench-GRN corresponds directly to the difficulty levels, reflecting progressively sharper and more nonlinear parameter regimes. *Easy* tasks exhibit quasi-linear responses where small perturbations reveal the graph clearly; *Medium* tasks introduce nonlinearity where saturation effects obscure weak edges; and *Hard* tasks feature bistability and switching behavior, where the graph is identifiable only through carefully designed multi-node perturbations. Appendix B lists the construction details, family definitions, filtering rules, and tasks.

Table 3: **Quantitative performance comparison** across the full benchmarks. *ED* denotes experiment design and *B* the experiment budget. For **NewtonBench** and **ActiveSciBench-Chem**: SA = Symbolic Accuracy (%), Ex. = Exact Accuracy (%), and RMSLE = Root Mean Squared Log Error. For **ActiveSciBench-GRN**: F1 = Edge F1 (%), Ex. = Exact Graph Accuracy (%), and Sign = Sign Accuracy (%). Best results are in **bold**; second-best results are underlined. LLM-SR and SR-Scientist apply only to equation-discovery benchmarks.

Method	ED	B	Easy			Medium			Hard			Overall		
			SA(↑)	Ex.(↑)	RMSLE(↓)	SA(↑)	Ex.(↑)	RMSLE(↓)	SA(↑)	Ex.(↑)	RMSLE(↓)	SA(↑)	Ex.(↑)	RMSLE(↓)
NewtonBench														
PySR	×	20	38.89	88.89	0.030	27.78	75.00	0.125	5.56	59.72	**0.398**	24.07	74.54	0.182
Bayesian Optimization	✓	20	40.28	84.72	0.070	27.78	70.83	0.212	5.56	50.00	0.730	24.54	68.52	0.334
Bayesian Experimental Design	✓	20	19.44	78.57	0.307	11.11	66.18	0.449	2.78	45.31	1.010	11.11	63.86	0.577
LLM-only (GPT-4o-mini)	✓	20	16.67	19.44	3.572	2.78	2.78	4.738	0.00	0.00	6.757	6.48	7.41	5.039
LLM-only (Claude Sonnet 4.6)	✓	20	64.31	78.24	0.783	60.39	73.56	0.791	58.83	**72.91**	0.818	61.18	74.90	0.797
Code-assisted (GPT-4o-mini)	✓	20	13.89	19.44	4.372	5.56	11.11	5.437	2.78	0.00	4.897	7.41	10.19	4.912
Code-assisted (Claude Sonnet 4.6)	✓	20	62.12	74.08	0.901	61.28	70.83	0.980	**59.83**	68.42	1.034	61.08	71.11	0.972
LLM-SR + EIG	✓	100	20.12	74.57	0.381	12.25	63.32	0.488	3.31	47.19	0.983	11.89	61.69	0.617
SR-Scientist + EIG	✓	100	22.38	76.12	0.379	14.82	66.84	0.445	4.84	50.41	0.843	14.01	64.46	0.556
LLM-AutoSciLab (Ours)	✓	20	**79.20**	**93.10**	**0.018**	**72.20**	**84.70**	**0.040**	51.40	66.70	0.404	**67.60**	**81.50**	**0.150**
ActiveSciBench-Chem														
PySR	×	60	33.33	33.33	0.455	0.00	11.54	1.545	0.00	0.00	0.601	7.89	15.79	1.212
Bayesian Optimization	✓	60	22.22	14.29	2.001	0.00	3.85	1.338	0.00	0.00	6.564	5.26	6.98	1.960
Bayesian Experimental Design	✓	60	**77.78**	77.78	**0.028**	19.23	30.77	0.333	0.00	0.00	0.187	31.58	39.47	0.249
LLM-only (GPT-4o-mini)	✓	60	0.00	0.00	0.563	0.00	0.00	0.777	0.00	0.00	0.849	0.00	0.00	0.764
LLM-only (Claude Sonnet 4.6)	✓	60	33.33	66.67	0.481	6.34	8.48	0.595	0.00	0.00	0.723	8.27	14.54	0.624
Code-assisted (GPT-4o-mini)	✓	60	0.00	0.00	0.554	0.00	0.00	0.869	0.00	0.00	0.894	0.00	0.00	0.824
Code-assisted (Claude Sonnet 4.6)	✓	60	22.22	33.33	0.412	4.82	5.48	0.538	0.00	0.00	0.593	5.79	7.86	0.538
LLM-SR + EIG	✓	100	55.56	34.81	0.881	14.73	23.28	1.092	0.00	0.00	4.817	15.75	16.52	2.431
SR-Scientist + EIG	✓	100	55.56	31.29	0.843	16.38	28.12	0.935	0.00	0.00	4.558	16.53	18.26	2.255
LLM-AutoSciLab (Ours)	✓	60	55.56	**88.89**	0.298	**22.22**	**37.04**	**0.179**	**42.86**	**52.38**	**0.154**	**35.09**	**50.88**	**0.189**
			F1(↑)	**Ex.**(↑)	**Sign**(↑)	**F1**(↑)	**Ex.**(↑)	**Sign**(↑)	**F1**(↑)	**Ex.**(↑)	**Sign**(↑)	**F1**(↑)	**Ex.**(↑)	**Sign**(↑)
ActiveSciBench-GRN														
GENIE3	×	20	41.54	0.00	85.33	35.00	0.00	71.89	39.27	0.00	80.00	38.60	0.00	79.07
GIES	×	20	**70.12**	20.00	97.78	47.07	0.00	80.56	51.61	0.00	79.00	56.27	6.67	85.78
NOTEARS	×	20	31.52	0.00	66.67	25.71	0.00	56.67	25.57	6.67	51.11	27.60	2.22	58.15
Random sampling	✓	20	33.64	0.00	86.67	33.46	0.00	86.67	39.58	6.67	80.00	35.56	2.22	84.44
Uncertainty sampling	✓	20	55.26	13.33	93.33	49.63	0.00	86.67	45.40	0.00	84.44	50.10	4.44	88.15
LLM-only (GPT-4o-mini)	✓	20	53.61	0.00	81.67	46.12	0.00	77.78	51.51	0.00	78.89	50.41	0.00	79.44
LLM-only (Claude Sonnet 4.6)	✓	20	59.88	0.00	93.33	64.27	0.00	93.33	53.81	0.00	90.00	59.32	0.00	92.22
Code-assisted (GPT-4o-mini)	✓	20	51.73	0.00	82.78	57.65	0.00	91.67	54.63	0.00	83.33	54.67	0.00	85.98
Code-assisted (Claude Sonnet 4.6)	✓	20	61.35	0.00	96.66	68.23	0.00	96.67	60.21	0.00	90.00	63.26	0.00	94.44
LLM-AutoSciLab (Ours)	✓	20	66.27	**26.67**	**100.0**	**83.13**	**40.00**	**100.0**	**68.06**	**26.67**	**94.44**	**72.49**	**31.11**	**98.15**

5 Experiments

5.1 Experimental Setup

Datasets. We evaluate LLM-AutoSciLab in a closed-loop scientific discovery setting where the system iteratively designs experiments and refines hypotheses under a fixed oracle query budget. Our study includes both quantitative comparisons with prior methods and targeted ablations. Specifically, we conduct experiments on: **NewtonBench**, spanning 12 physics domains across multiple difficulty levels and variants; **ActiveSciBench-Chem**, a suite of compositional enzyme kinetics tasks; and **ActiveSciBench-GRN**, which focuses on graph-structured discovery in gene regulatory networks. For ActiveSciBench-Chem and NewtonBench, we report: (i) symbolic accuracy, (ii) predictive error via RMSLE, (iii) numerical exact accuracy. Symbolic accuracy is stricter than numerical accuracy, as approximate fits may not recover the true form. For ActiveSciBench-GRN (structure discovery), we evaluate structural recovery using edge-level precision, recall, F1, and sign accuracy (activation vs. repression). Additional metric details are provided in Appendix C.

Baselines. We compare against a broad set of baselines: **symbolic regression methods** such as PySR [14] on fixed datasets without experiment design, and **active learning methods** (Bayesian Optimization and Bayesian Experimental Design) that select experiments but do not model symbolic structure. We also evaluate active-acquisition variants of LLM-SR [44] and SR-Scientist [51], together with LLM-only prompting and code-assisted LLMs. For ActiveSciBench-GRN, we additionally evaluate **graph discovery baselines** GENIE3 [25], GIES [23], and NOTEARS [53] (offline), as well as Random and Uncertainty sampling. For ablations and diagnostic analyses, we use stratified, representative subsets to improve computational tractability (Appendix E.3). Unless explicitly labeled as Claude Sonnet 4.6, LLM baselines and LLM-AutoSciLab use `GPT-4o-mini` as the primary model; Claude Sonnet 4.6 is accessed through Amazon Bedrock on AWS. LLM-AutoSciLab additionally uses `Qwen2.5-7B-Instruct` as the smaller local ensemble model. Appendix A.5 reports more results.

5.2 Main Results

As shown in Table 3, we evaluate LLM-AutoSciLab using `GPT-4o-mini` across NewtonBench, ActiveSciBench-Chem, and ActiveSciBench-GRN under fixed oracle budgets. Across all three benchmarks, LLM-AutoSciLab achieves the strongest overall recovery, but the source of advantage differs

by setting: NewtonBench tests symbolic identifiability with known variables, ActiveSciBench-Chem requires discovering relevant kinetic variables, and ActiveSciBench-GRN requires recovering graph structure from sparse interventions. Appendix A.2 contains experiments using `Qwen-3` family LLMs.

NewtonBench. NewtonBench isolates symbolic recovery when relevant variables are known. Strong fitting baselines such as PySR and Bayesian Optimization achieve high exact accuracy (74.54%, 68.52%) but low symbolic accuracy (24.07%, 24.54%), indicating overfitting to observed regimes without recovering the underlying law. The GPT-4o-mini LLM-only and code-assisted baselines remain below 8% SA, while their Claude Sonnet 4.6 counterparts improve to approximately 61% SA but remain below LLM-AutoSciLab. Active LLM-SR and SR-Scientist variants also remain below 15% SA despite receiving an experiment budget of 100, compared with 20 for LLM-AutoSciLab. In contrast, LLM-AutoSciLab achieves 67.60% symbolic accuracy and 81.50% exact accuracy. This gap highlights that *hypothesis-conditioned experimentation improves structural identification rather than merely numerical fit*.

ActiveSciBench-Chem. ActiveSciBench-Chem requires identifying both relevant variables and kinetic structure through interaction with the experimental interface. BED is the strongest conventional baseline (31.58% SA), performing well on easy tasks (77.78% SA) but collapsing on hard settings (0.00% SA/Ex.), suggesting limited coverage beyond the candidate model class. GPT-4o-mini LLM-only and code-assisted baselines achieve 0.00% SA/Ex.; Claude Sonnet 4.6 improves their overall SA to 8.27% and 5.79%, respectively, while the active LLM-SR and SR-Scientist variants reach 15.75% and 16.53% SA at budget 100. These methods often produce plausible rate laws but tend to default to generic templates rather than resolving variable relevance and kinetic family. In contrast, LLM-AutoSciLab achieves the best overall performance (35.09% SA, 50.88% Ex.) and remains robust on hard tasks (42.86% SA, 52.38% Ex.). The results suggest that *LLM-generated hypotheses enable exploration beyond fixed mechanism libraries, which is critical for nonstandard kinetics*.

ActiveSciBench-GRN. ActiveSciBench-GRN evaluates graph-structured mechanism recovery. Offline methods recover partial structure but rarely the full graph: GIES achieves 56.27% F1 but only 6.67% exact accuracy. Active baselines improve edge recovery (e.g., uncertainty sampling: 50.10% F1) but still fail to recover full graphs (4.44% Ex.). Claude Sonnet 4.6 improves the LLM-only and code-assisted baselines to 59.32% and 63.26% edge F1, respectively, but neither recovers a complete graph. LLM-AutoSciLab achieves 72.49% F1, 31.11% exact graph accuracy, and 98.15% sign accuracy. We observe that most methods perform very well on the sign metric, indicating that it is easier to determine whether a node suppresses, activates, or has no effect on other nodes. This demonstrates that *accurate graph recovery requires targeted, hypothesis-driven perturbations that disambiguate competing signed structures*, rather than passive fitting or uncertainty-based sampling.

6 Analysis

6.1 How does the closed loop refine hypotheses?

We next examine how the closed-loop design of LLM-AutoSciLab contributes to its performance. We remove individual feedback mechanisms and include matched open-loop controls. *One-shot passive* collects all observations upfront and runs the hypothesis and refinement pipeline once. *Open-loop plan* chooses all experiments from the initial static hypotheses, without adapting the plan to their outcomes. *No acquisition feedback* observes each result but chooses subsequent experiments without conditioning on the current hypotheses. *No hypothesis revision* retains adaptive acquisition but does not revise the hypothesis space using new evidence. All variants use the same discovery objective, oracle budget, LLM backbones, and refinement backend. Table 4 summarizes the results. The full method performs best overall, although the effect of each feedback pathway differs across benchmarks. ActiveSciBench-Chem is particularly sensitive to one-shot inference and open-loop planning, indicating that competing kinetic mechanisms require experiments chosen in response to earlier outcomes. ActiveSciBench-GRN degrades most without acquisition feedback, consistent with graph recovery requiring targeted interventions that distinguish direct, mediated, and feedback effects. NewtonBench degrades under every control, showing that both adaptive experiment selection

Table 4: **Closed-loop discovery ablation.** SA is symbolic accuracy; Ex. is exact graph accuracy.

Variant	Newton SA	Chem SA	GRN Ex.
LLM-AutoSciLab	**54.17**	**33.33**	**38.00**
One-shot passive	41.67	0.00	30.00
Open-loop plan	50.00	0.00	21.00
No acquisition feedback	41.67	21.85	16.00
No hypothesis revision	37.50	16.67	34.00

and repeated hypothesis revision contribute to symbolic recovery. Overall, the results show that the improvement comes from coupling hypothesis-based data acquisition and refinement, showing that the closed-loop nature of LLM-AutoSciLab significantly contributes to its performance.

6.2 Qualitative Analysis

Figure 3 presents a qualitative NewtonBench case study illustrating failure modes under limited-budget mechanism discovery. The target system contains two additive components: a restoring force term and a nonlinear damping term. Fit-driven baselines such as PySR and Bayesian optimization achieve low local error by introducing spurious or entangled terms, but fail to recover the correct mechanistic structure. In particular, PySR inserts an unnecessary sinusoidal component, while Bayesian optimization converges to numerically accurate but mechanistically incorrect expressions. LLM-only and code-assisted LLM methods instead collapse to simplified textbook harmonic forms, identifying only partial structure and missing the nonlinear damping behavior. Bayesian experiment design moves toward the correct components, but ultimately still converges to a fit-driven approximation. In contrast, LLM-AutoSciLab recovers the correct additive structure and closely matches the hidden constants, including a recovered coefficient of $0.4915 \approx 1/2$ and a damping exponent within roughly 1% of the ground truth, highlighting the importance of discriminative experimentation in accurate mechanism recovery.

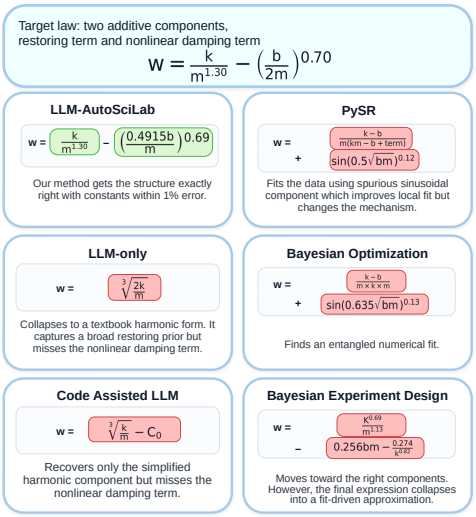


Figure 3: **Qualitative NewtonBench case study.** LLM-AutoSciLab recovers the correct symbolic structure, while other baselines introduce spurious terms, collapse to incorrect families, or recover only simplified harmonic forms.

6.3 Ablation Study

Figure 4 reports single-component removal results across all three benchmarks. Removing hypothesis-conditioned acquisition causes a substantial drop across all benchmarks, showing that candidate mechanisms must guide data acquisition under limited oracle budgets. Other components contribute differently depending on the source of difficulty. NewtonBench stresses sparse functional identification through counterfactual laws, making diverse hypothesis generation and mechanism stability important; without them, the LLM is drawn toward canonical physics laws. ActiveSciBench-Chem requires reasoning over the input space and identifying mechanistically relevant variables, making memory crucial for disambiguating competing mechanisms. ActiveSciBench-GRN relies on accumulating perturbation evidence, making evidence preservation and intervention selection more central. These ablation patterns also support the diversity of our suite: NewtonBench, ActiveSciBench-Chem, and ActiveSciBench-GRN are sensitive to different removed components, suggesting they probe complementary discovery capabilities rather than one shared fitting problem. Overall, the results show that LLM-AutoSciLab functions as a closed-loop pipeline across discovery settings, with each component contributing meaningfully.

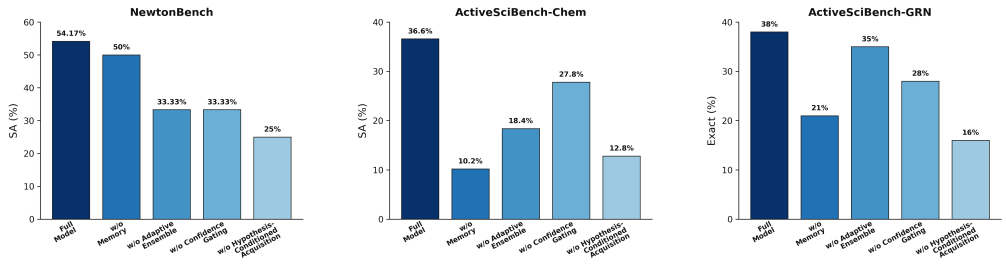


Figure 4: Ablation study across all benchmarks removing one component from LLM-AutoSciLab.

6.4 Experiment Budget Analysis

Figure 5 plots recovery versus query budget. We measure sample efficiency as the query budget each baseline needs to match LLM-AutoSciLab's fixed-budget performance, using symbolic accuracy for NewtonBench and ActiveSciBench-Chem and exact graph accuracy for ActiveSciBench-GRN. Comparison methods require 2.17–$2.92\times$ as many queries on NewtonBench, 2.36–$5.97\times$ on ActiveSciBench-Chem, and 1.3–$7.4\times$ on ActiveSciBench-GRN (Appendix A.3). By conditioning acquisition on competing hypotheses, each oracle call is more likely to resolve structural ambiguity across symbolic laws, kinetic rate mechanisms, and signed regulatory graphs. Together, these results show that *LLM-AutoSciLab is significantly more sample efficient than the baselines*.

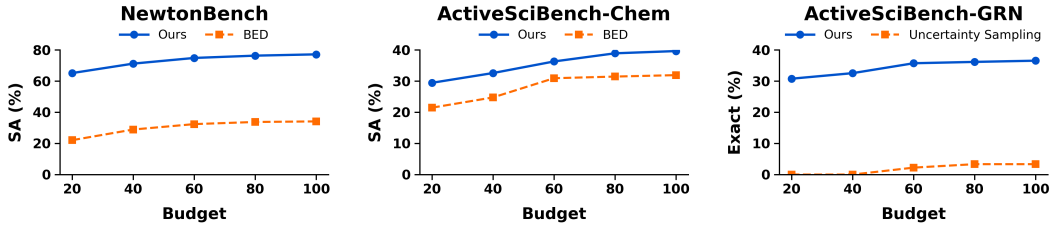


Figure 5: Budget ablations across benchmarks showing the recovery metric versus query budget.

7 Conclusion

We introduced LLM-AutoSciLab, a closed-loop algorithm for scientific discovery that formalizes discovery as iterative experimental design over an evolving hypothesis set. LLM-AutoSciLab instantiates a structured algorithmic loop that (i) *generates a diverse hypothesis set*, (ii) selects experiments via a *hypothesis-conditioned acquisition objective*, and (iii) refines candidates through *data-driven optimization with feedback*. This formulation shifts the objective of data acquisition from predictive uncertainty reduction to hypothesis discrimination, improving true mechanism recovery under limited experimental budgets. To overcome the lack of evaluation settings for closed-loop discovery, we introduce **ActiveSciBench-Chem** and **ActiveSciBench-GRN**, *benchmarks that recast scientific discovery as an active, budget-constrained process requiring joint experiment design, variable selection, and recovery of underlying mechanisms*, enabling systematic evaluation beyond static function fitting. Across NewtonBench, ActiveSciBench-Chem, and ActiveSciBench-GRN, LLM-AutoSciLab achieves higher symbolic and structural recovery with fewer queries than prior methods, demonstrating that *aligning data acquisition with hypothesis discrimination rather than predictive accuracy improves both efficiency and reliability of scientific discovery*.

Limitations. LLM-AutoSciLab uses simulator-based oracles, so physical-lab noise, failures, costs, and operational constraints are not fully modeled. Performance also depends on the quality of LLM-generated hypotheses and on the coverage of the parser and refinement backends. Broader domains, richer refinement tools, and real-world validation remain future work.

Acknowledgements

This research was partially supported by the U.S. National Science Foundation (NSF) under Grant No. 2416728 and Autodesk Research. We thank Amazon Web Services (AWS) for providing the computational resources necessary to conduct the LLM inference for this study through an AWS Cloud Research Grant. This work was supported by a Laboratory Directed Research & Development (LDRD) project. This work was performed at the Center for Integrated Nanotechnologies, a U.S. Department of Energy Office of Science user facility. An employee of National Technology & Engineering Solutions of Sandia, LLC under Contract No authored this article. DE-NA0003525 with the U.S. DOE. The employee retains all rights to the article and is solely responsible for its contents. The U.S. Government retains a non-exclusive, paid-up, irrevocable, worldwide license to publish or reproduce this work for government purposes. Public access will be provided in accordance with the DOE Public Access Plan: https://www.energy.gov/downloads/doe-public-access-plan.

References

[1] Nikhil Abhyankar, Sanchit Kabra, Saaketh Desai, and Chandan K. Reddy. LLEMA: Evolutionary search with LLMs for multi-objective materials discovery. In *The Fourteenth International Conference on Learning Representations*, 2026.

[2] Nikhil Abhyankar, Sha Li, Sanchit Kabra, Naren Ramakrishnan, Yulia Gel, and Chandan K. Reddy. LLM-ACES: Closed-loop discovery of dynamical systems with LLM-guided adaptive search. *arXiv preprint arXiv:2606.25039*, 2026.

[3] Milad Abolhasani and Eugenia Kumacheva. The rise of self-driving labs in chemical and materials sciences. *Nature Synthesis*, 2:483 – 492, 2023.

[4] Dhruv Agarwal, Bodhisattwa Prasad Majumder, Reece Adamson, Megha Chakravorty, Satvika Reddy Gavireddy, Aditya Parashar, Harshit Surana, Bhavana Dalvi Mishra, Andrew McCallum, Ashish Sabharwal, and Peter Clark. AutoDiscovery: Open-ended scientific discovery via Bayesian surprise. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

[5] Microsoft Research AI4Science and Microsoft Azure Quantum. The impact of large language models on scientific discovery: A preliminary study using GPT-4. *arXiv preprint arXiv:2311.07361*, 2023.

[6] Michael Bailey, Saeed Moayedpour, Ruijiang Li, Alejandro Corrochano-Navarro, Alexander Kötter, Lorenzo Kogler-Anele, Saleh Riahi, Christoph Grebner, Gerhard Hessler, Hans Matter, Marc Bianciotto, Pablo Mas, Ziv Bar-Joseph, and Sven Jager. Deep batch active learning for drug discovery. *eLife*, January 2024.

[7] Pouya Behzadifar, Parshin Shojaee, Sanchit Kabra, Kazem Meidani, and Chandan K. Reddy. Decompose, adapt, and evolve: Towards efficient scientific equation discovery with large language models. In *NeurIPS 2025 AI for Science Workshop*, 2025.

[8] George E. P. Box and William J. Hill. Discrimination among mechanistic models. *Technometrics*, 9(1):57–71, 1967.

[9] Qiguang Chen, Mingda Yang, Libo Qin, Jinhao Liu, Zheng Yan, Jiannan Guan, Dengyun Peng, Yiyan Ji, Hanjing Li, Mengkang Hu, et al. Ai4research: A survey of artificial intelligence for scientific research. *arXiv preprint arXiv:2507.01903*, 2025.

[10] Tingting Chen, Beibei Lin, Zifeng Yuan, Qiran Zou, Hongyu He, Anirudh Goyal, Yew-Soon Ong, and Dianbo Liu. HypoSpace: Evaluating LLM creativity as set-valued hypothesis generators under underdetermination. *arXiv preprint arXiv:2510.15614*, 2025.

[11] Yimeng Chen, Piotr Piękos, Mateusz Ostaszewski, Firas Laakom, and Jürgen Schmidhuber. PhysGym: Benchmarking LLMs in interactive physics discovery with controlled priors. *Advances in Neural Information Processing Systems*, 38, 2025.

[12] Mathieu Chevalley, Yusuf H Roohani, Arash Mehrjou, Jure Leskovec, and Patrick Schwab. A large-scale benchmark for network inference from single-cell perturbation data. *Communications Biology*, 8(1):412, 2025.

[13] Deepro Choudhury, Sinead Williamson, Adam Golinski, Ning Miao, Freddie Bickford Smith, Michael Kirchhof, Yizhe Zhang, and Tom Rainforth. BED-LLM: Intelligent information gathering with LLMs and Bayesian experimental design. In *International Conference on Learning Representations*, pages 84375–84405, 2026.

[14] Miles Cranmer. Interpretable machine learning for science with PySR and SymbolicRegression.jl. *arXiv preprint arXiv:2305.01582*, 2023.

[15] Stéphane d'Ascoli, Sören Becker, Philippe Schwaller, Alexander Mathis, and Niki Kilbertus. ODEFormer: Symbolic regression of dynamical systems with transformers. In *The Twelfth International Conference on Learning Representations*, 2024.

[16] Saaketh Desai, Sadhvikas Addamane, Jeffrey Y Tsao, Igal Brener, Laura P Swiler, Remi Dingreville, and Prasad P Iyer. AutoSciLab: A self-driving laboratory for interpretable scientific discovery. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 146–154, 2025.

[17] Jingru Gan, Peichen Zhong, Yuanqi Du, Yanqiao Zhu, Chenru Duan, Haorui Wang, Daniel Schwalbe-Koda, Carla P Gomes, Kristin A Persson, and Wei Wang. MatLLMSearch: Crystal structure discovery with evolution-guided large language models. *arXiv preprint arXiv:2502.20933*, 2025.

[18] Kanishk Gandhi, Michael Y Li, Lyle Goodyear, Agam Bhatia, Louise Li, Aditi Bhaskar, Mohammed Zaman, and Noah D Goodman. BoxingGym: Benchmarking progress in automated experimental design and model discovery. *arXiv preprint arXiv:2501.01540*, 2025.

[19] Ali Essam Ghareeb, Benjamin Chang, Ludovico Mitchener, Angela Yiu, Caralyn J Szostkiewicz, Dmytro Shved, Gavin J Gyimesi, Jon M Laurent, Samantha M Wright, Muhammed T Razzak, et al. A multi-agent system for automating scientific discovery. *Nature*, pages 1–3, 2026.

[20] Juraj Gottweis, Wei-Hung Weng, Alexander Daryin, Tao Tu, Anil Palepu, Petar Sirkovic, Artiom Myaskovsky, Felix Weissenberger, Keran Rong, Ryutaro Tanno, et al. Towards an ai co-scientist. *arXiv preprint arXiv:2502.18864*, 2025.

[21] Arya Grayeli, Atharva Sehgal, Omar Costilla-Reyes, Miles Cranmer, and Swarat Chaudhuri. Symbolic regression with a learned concept library. *Advances in Neural Information Processing Systems*, 37:44678–44709, 2024.

[22] Florian Häse, Matteo Aldeghi, Riley J Hickman, Loïc M Roch, Melodie Christensen, Elena Liles, Jason E Hein, and Alán Aspuru-Guzik. Olympus: a benchmarking framework for noisy optimization and experiment planning. *Machine Learning: Science and Technology*, 2(3):035021, 2021.

[23] Alain Hauser and Peter Bühlmann. Characterization and greedy learning of interventional markov equivalence classes of directed acyclic graphs. *The Journal of Machine Learning Research*, 13(1):2409–2464, 2012.

[24] Kexin Huang, Romain Lopez, Jan-Christian Hütter, Takamasa Kudo, Antonio Rios, and Aviv Regev. Sequential optimal experimental design of perturbation screens guided by multi-modal priors. *bioRxiv*, 2023.

[25] Vân Anh Huynh-Thu, Alexandre Irrthum, Louis Wehenkel, and Pierre Geurts. Inferring regulatory networks from expression data using tree-based methods. *PLoS ONE*, 5, 2010.

[26] Peter Jansen, Peter Clark, Doug Downey, and Daniel S Weld. Generating literature-driven scientific theories at scale. *arXiv preprint arXiv:2601.16282*, 2026.

[27] Nan Jiang, Md Nasim, and Yexiang Xue. Active symbolic discovery of ordinary differential equations via phase portrait sketching. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 17626–17634, 2025.

[28] Sanchit Kabra, Shobhnik Kriplani, Parshin Shojaee, and Chandan K. Reddy. SURFACEBENCH: A geometry-aware benchmark for symbolic surface discovery. *Transactions on Machine Learning Research*, 2026.

[29] A Gilad Kusne, Heshan Yu, Changming Wu, Huairuo Zhang, Jason Hattrick-Simpers, Brian DeCost, Suchismita Sarker, Corey Oses, Cormac Toher, Stefano Curtarolo, et al. On-the-fly closed-loop materials discovery via Bayesian active learning. *Nature communications*, 11(1):5966, 2020.

[30] Gregory W. Kyro, Anton Morgunov, Rafael I. Brent, and Victor S. Batista. Chemspaceal: An efficient active learning methodology applied to protein-specific molecular generation. *Journal of Chemical Information and Modeling*, 64(3):653–665, January 2024.

[31] Pat Langley. Integrated systems for computational scientific discovery. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(20):22598–22606, Mar. 2024.

[32] Julia Ling, Maxwell Hutchinson, Erin Antono, Sean Paradiso, and Bryce Meredig. High-dimensional materials and process optimization using data-driven experimental design with well-calibrated uncertainty estimates. *Integrating Materials and Manufacturing Innovation*, 6(3):207–217, 2017.

[33] B. P. MacLeod, F. G. L. Parlane, T. D. Morrissey, F. Häse, L. M. Roch, K. E. Dettelbach, R. Moreira, L. P. E. Yunker, M. B. Rooney, J. R. Deeth, V. Lai, G. J. Ng, H. Situ, R. H. Zhang, M. S. Elliott, T. H. Haley, D. J. Dvorak, A. Aspuru-Guzik, J. E. Hein, and C. P. Berlinguette. Self-driving laboratory for accelerated discovery of thin-film materials. *Science Advances*, 6(20):eaaz8867, 2020.

[34] Bodhisattwa Prasad Majumder, Harshit Surana, Dhruv Agarwal, Sanchaita Hazra, Ashish Sabharwal, and Peter Clark. Data-driven discovery with large generative models. *arXiv preprint arXiv:2402.13610*, 2024.

[35] Alexey A. Melnikov, Hendrik Poulsen Nautrup, Mario Krenn, Vedran Dunjko, Markus Tiersch, Anton Zeilinger, and Hans J. Briegel. Active learning machine learns to create new quantum experiments. *Proceedings of the National Academy of Sciences*, 115(6):1221–1226, 2018.

[36] Long Ouyang, Michael Henry Tessler, Daniel Ly, and Noah Goodman. Practical optimal experiment design with probabilistic programs. *arXiv preprint arXiv:1608.05046*, 2016.

[37] Brenden K Petersen, Mikel Landajuela Larma, Terrell N. Mundhenk, Claudio Prata Santiago, Soo Kyung Kim, and Joanne Taery Kim. Deep symbolic regression: Recovering mathematical expressions from data via risk-seeking policy gradients. In *International Conference on Learning Representations*, 2021.

[38] Aditya Pratapa, Amogh P. Jalihal, Jeffrey N. Law, Aditya Bharadwaj, and T. M. Murali. Benchmarking algorithms for gene regulatory network inference from single-cell transcriptomic data. *Nature Methods*, 17(2):147–154, 2020.

[39] Jason Qin, Hans-Hermann Wessels, Carlos Fernandez-Granda, and Yuhan Hao. Active learning for efficient discovery of optimal gene combinations in the combinatorial perturbation space. In *NeurIPS 2024 Workshop on AI for New Drug Modalities*, 2024.

[40] Chandan K. Reddy and Parshin Shojaee. Towards scientific discovery with generative ai: Progress, opportunities, and challenges. In *Proceedings of the AAAI conference on artificial intelligence*, volume 39, pages 28601–28609, 2025.

[41] Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Matej Balog, M Pawan Kumar, Emilien Dupont, Francisco JR Ruiz, Jordan S Ellenberg, Pengming Wang, Omar Fawzi, et al. Mathematical discoveries from program search with large language models. *Nature*, 625(7995):468–475, 2024.

[42] Yusuf Roohani, Andrew Lee, Qian Huang, Jian Vora, Zachary Steinhart, Kexin Huang, Alexander Marson, Percy Liang, and Jure Leskovec. BioDiscoveryAgent: An AI agent for designing genetic perturbation experiments. In *International Conference on Learning Representations*, pages 26417–26466, 2025.

[43] Thomas Schaffter, Daniel Marbach, and Dario Floreano. GeneNetWeaver: in silico benchmark generation and performance profiling of network inference methods. *Bioinformatics*, 27(16):2263–2270, 08 2011.

[44] Parshin Shojaee, Kazem Meidani, Shashank Gupta, Amir Barati Farimani, and Chandan K. Reddy. LLM-SR: Scientific equation discovery via programming with large language models. In *The Thirteenth International Conference on Learning Representations*, 2025.

[45] Parshin Shojaee, Ngoc-Hieu Nguyen, Kazem Meidani, Amir Barati Farimani, Khoa D Doan, and Chandan K. Reddy. LLM-SRBench: A new benchmark for scientific equation discovery with large language models. In *Forty-second International Conference on Machine Learning*, 2025.

[46] Makoto Takamoto, Timothy Praditia, Raphael Leiteritz, Daniel MacKinlay, Francesco Alesiani, Dirk Pflüger, and Mathias Niepert. PDEBench: An extensive benchmark for scientific machine learning. *Advances in neural information processing systems*, 35:1596–1611, 2022.

[47] Silviu-Marian Udrescu and Max Tegmark. AI Feynman: A physics-inspired method for symbolic regression. *Science advances*, 6(16):eaay2631, 2020.

[48] Hanchen Wang, Tianfan Fu, Yuanqi Du, Wenhao Gao, Kexin Huang, Ziming Liu, Payal Chandak, Shengchao Liu, Peter Van Katwyk, Andreea Deac, et al. Scientific discovery in the age of artificial intelligence. *Nature*, 620(7972):47–60, 2023.

[49] Haorui Wang, Marta Skreta, Cher Tian Ser, Wenhao Gao, Lingkai Kong, Felix Strieth-Kalthoff, Chenru Duan, Yuchen Zhuang, Yue Yu, Yanqiao Zhu, Yuanqi Du, Alan Aspuru-Guzik, Kirill Neklyudov, and Chao Zhang. Efficient evolutionary search over chemical space with large language models. In *The Thirteenth International Conference on Learning Representations*, 2025.

[50] Matt L Wiemann, Lindsay M Smith, Peter Melchior, Siddharth Mishra-Sharma, Andrew Gordon Wilson, Pavel Izmailov, and Carolina Cuesta-Lázaro. DiscoverPhysics: Benchmarking LLMs for out-of-the-box scientific thinking. *arXiv preprint arXiv:2605.26087*, 2026.

[51] Shijie Xia, Yuhan Sun, and Pengfei Liu. SR-Scientist: Scientific equation discovery with agentic AI. In *International Conference on Learning Representations*, pages 75787–75811, 2026.

[52] Tianshi Zheng, Kelvin Kiu Wai Tam, Newt Nguyen Kim Hue Nam, Baixuan Xu, Zhaowei Wang, Cheng Jiayang, Hong Ting Tsang, Weiqi Wang, Jiaxin Bai, Tianqing Fang, Yangqiu Song, Ginny Wong, and Simon See. NewtonBench: Benchmarking generalizable scientific law discovery in LLM agents. In *The Fourteenth International Conference on Learning Representations*, 2026.

[53] Xun Zheng, Bryon Aragam, Pradeep K Ravikumar, and Eric P Xing. DAGs with NO TEARS: Continuous optimization for structure learning. *Advances in neural information processing systems*, 31, 2018.

[54] Yangqiaoyu Zhou, Haokun Liu, Tejes Srivastava, Hongyuan Mei, and Chenhao Tan. Hypothesis generation with large language models. In *Proceedings of the 1st Workshop on NLP for Science (NLP4Science)*, pages 117–139, 2024.

Impact Statement

LLM-AutoSciLab accelerates scientific discovery by automating hypothesis-driven experimentation, with potential benefits for researchers in biology, chemistry, and physics who face costly experimental budgets, reducing the number of experiments needed to recover governing mechanisms and lowering barriers for smaller research groups. The primary risks are over-reliance on model outputs in safety-critical domains such as drug discovery, where a plausible but incorrect mechanistic law could have downstream consequences, and the inheritance of LLM biases that may systematically favor well-represented mechanisms over genuinely novel ones. The framework currently targets simulator-based discovery rather than direct laboratory deployment, limiting immediate risk, but domain-specific safety review remains essential before application in sensitive real-world contexts. We do not anticipate direct dual-use concerns.

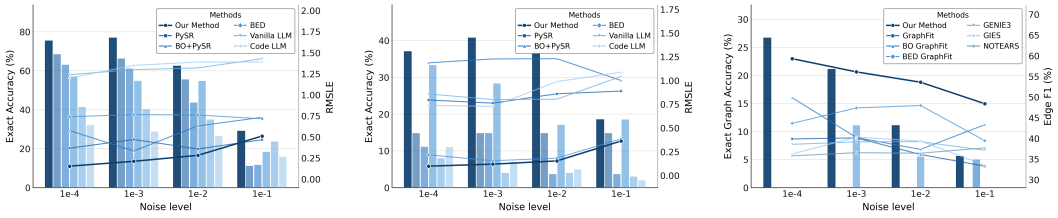
Reproducibility statement

We provide implementation and experimental details throughout the paper, including the methodology in Section 3 and Appendix E.2, dataset construction details in Appendix B, and metric definitions in Appendix C. The code and data accompanying the benchmark are intended to support independent reproduction.

Appendix

A Additional Results

A.1 Noise Sensitivity



(a) NewtonBench (b) ActiveSciBench-Chem (c) ActiveSciBench-GRN

Figure 6: Robustness to observation noise across (a) NewtonBench, (b) ActiveSciBench-Chem, and (c) ActiveSciBench-GRN. Bars report exact accuracy, while lines report the continuous error or graph recovery metric: RMSLE for NewtonBench and ActiveSciBench-Chem, and Edge F1 for ActiveSciBench-GRN.

Figure 6 reports a noise sensitivity analysis evaluating robustness to increasing levels of observation noise across all three benchmarks. For each benchmark, we inject controlled Gaussian noise at varying levels and measure the effect on both primary recovery metrics and predictive fidelity. Across all settings, LLM-AutoSciLab shows stronger recovery than baselines as noise increases, consistent with scientific priors that constrain the search to physically plausible structures even as data quality degrades. On NewtonBench and ActiveSciBench-Chem, symbolic accuracy exhibits threshold behavior, degrading in a step-function pattern while RMSLE increases smoothly, reflecting a transition between a regime where noise blurs parameter estimates but structural identifiability is preserved, and a regime where competing hypotheses become observationally indistinguishable within the current budget. On ActiveSciBench-GRN, the two recovery metrics decouple sharply under noise. Exact graph accuracy drops severely even at moderate noise levels, while edge F1 deteriorates slowly and gradually. This dissociation reveals that noise primarily disrupts precise topology recovery while the broader edge-level structure remains partially identifiable. The result suggests that activation versus repression provides a stronger, more noise-resilient signal than exact graph structure and points to precise topology recovery as the primary fragility of the current framework under noisy perturbation settings.

A.2 Model Capability

Table 5 evaluates the effect of LLM backbone capability on closed-loop discovery. We keep the `Qwen2.5-7B-Instruct` as the smaller local ensemble model, but vary the primary model driving the experiments. Model scale is most beneficial on the more compositional and structured benchmarks. ActiveSciBench-Chem improves consistently from `Qwen3-4B` to `Qwen3-32B` across SA, exact accuracy, and RMSLE, while ActiveSciBench-GRN shows clear gains in edge F1 and exact graph accuracy. This suggests that stronger backbones provide better mechanistic priors for selecting relevant variables, proposing discriminative experiments, and reasoning over structured mechanisms. NewtonBench reveals a different pattern. `Qwen3-32B` achieves the lowest RMSLE and highest exact accuracy, but `Qwen3-14B` obtains higher symbolic accuracy. Thus, larger models can fit behavior more accurately without always recovering the exact symbolic form. Since the difference is small, this should not be interpreted as evidence against larger models; rather, the non-monotonic symbolic trend suggests that closed-loop discovery is not determined by scale alone, but by the interaction between hypothesis generation, experiment selection, and refinement.

Table 5: **LLM backbone comparison.** GPT-4o-mini is the default backbone; Qwen3 models evaluate open-weight scaling.

	Newton			Chem			GRN		
Backbone	**SA**	**Ex.**	**RMSLE**	**SA**	**Ex.**	**RMSLE**	**F1**	**Ex.**	**Sign**
GPT-4o-mini	**67.60**	81.50	0.150	**35.09**	50.88	0.189	**72.49**	**31.11**	98.15
Qwen3-4B	57.78	82.86	0.174	12.92	53.56	0.3766	49.56	21.78	93.52
Qwen3-14B	54.29	86.11	0.127	23.88	50.59	0.1574	51.19	22.34	92.16
Qwen3-32B	60.56	**88.12**	**0.101**	25.56	**52.81**	**0.0815**	62.70	25.88	**98.70**

A.3 Relative Sample Efficiency

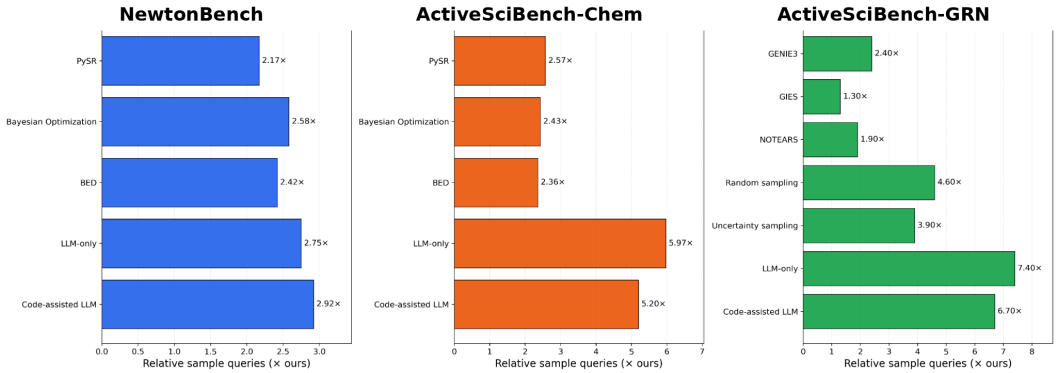


Figure 7: **Relative sample efficiency across benchmarks.** Each bar shows the multiplicative number of oracle queries required by a comparison method to match the fixed-budget performance of LLM-AutoSciLab (lower is better), measured using symbolic accuracy for NewtonBench and ActiveSciBench-Chem and exact graph accuracy for ActiveSciBench-GRN. Reference budgets are B=20, B=60, and B=20, respectively. LLM-only and Code-assisted LLM denote the GPT-4o-mini variants.

Figure 7 reports target-matching relative sample efficiency across the three benchmark families. For each benchmark, we treat the fixed-budget performance of LLM-AutoSciLab as the reference target and measure how many oracle queries a comparison method requires to match that target on the benchmark's recovery metric: symbolic accuracy for NewtonBench and ActiveSciBench-Chem, and exact graph accuracy for ActiveSciBench-GRN. A relative sample efficiency of $2\times$ therefore means that the comparison method requires twice as many queries as LLM-AutoSciLab to attain the same level of recovery.

On NewtonBench, all comparison methods require more than twice the query budget of LLM-AutoSciLab, with symbolic regression baselines ranging from $2.17\times$ to $2.58\times$ and the GPT-4o-mini LLM-only variants ranging from $2.75\times$ to $2.92\times$. On ActiveSciBench-Chem, the non-LLM methods require $2.36\times$–$2.57\times$, while the GPT-4o-mini LLM-only and code-assisted conditions require $5.97\times$ and $5.20\times$, respectively. On ActiveSciBench-GRN, GENIE3, GIES, and NOTEARS require $2.4\times$, $1.3\times$, and $1.9\times$, while active and GPT-4o-mini LLM alternatives require $3.9\times$–$7.4\times$. Taken together, these results show that the main benefit of LLM-AutoSciLab is not only improved final recovery but also substantially better budget utilization, since conditioning acquisition on explicitly competing mechanistic explanations makes each oracle query more informative for resolving structural ambiguity.

A.4 Qualitative Analysis

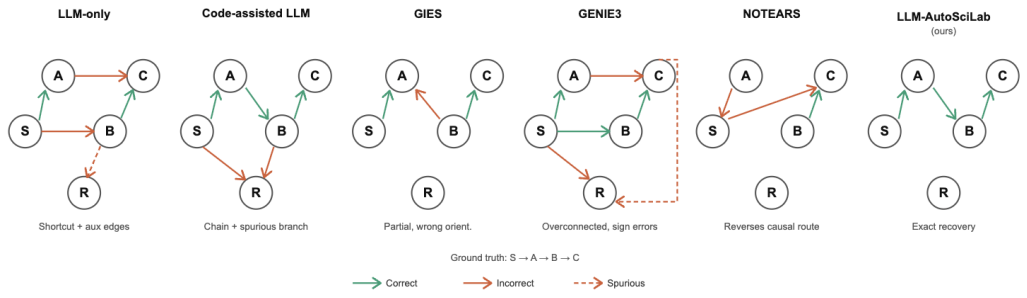


Figure 8: **Qualitative ActiveSciBench-GRN case study.** LLM-AutoSciLab exactly recovers the sparse activation chain, while baselines either add spurious auxiliary edges, reverse edge orientation, or recover only partial structure. Green edges indicate correctly recovered relations; red edges indicate incorrect or spurious relations.

Figure 8 shows a representative ActiveSciBench-GRN example where the target mechanism is a sparse activation chain, $S \to A \to B \to C$, with an irrelevant regulator R. LLM-AutoSciLab exactly recovers the ground-truth chain and correctly excludes R, indicating that its perturbation choices isolate the causal backbone rather than merely fitting correlated responses. In contrast, the baselines recover only parts of the structure. LLM-only adds shortcut and auxiliary edges, code-assisted LLM preserves the main chain but introduces a spurious side branch, and classical graph-discovery methods either overconnect the graph, reverse orientations, or miss key dependencies. This example illustrates the main GRN failure mode: baselines often detect local associations but fail to distinguish direct causal edges from indirect or irrelevant perturbation effects, whereas hypothesis-conditioned acquisition supports targeted disambiguation of the signed graph structure.

A.5 Statistical significance

Table 6: **Standard Deviation across benchmarks.** Values are reported as mean ± standard deviation across 3 seeds. For NewtonBench and ActiveSciBench-Chem, columns are SA / Exact / RMSLE. For ActiveSciBench-GRN, columns are F1 / Exact / Sign.

Method	SA	Ex.	RMSLE
	NewtonBench		
PySR	24.07 ± 0.00	74.54 ± 1.96	0.182 ± 0.008
BO+PySR	24.54 ± 0.65	68.52 ± 2.62	0.334 ± 0.023
BED	11.11 ± 0.00	63.86 ± 1.09	0.577 ± 0.044
LLM-only	6.48 ± 0.40	7.41 ± 0.80	5.039 ± 0.156
Code-assisted LLM	7.41 ± 0.50	10.19 ± 0.20	4.912 ± 0.128
LLM-AutoSciLab (Ours)	**67.60 ± 0.20**	**81.50 ± 1.30**	**0.150 ± 0.003**
	ActiveSciBench-Chem		
PySR	7.89 ± 0.00	15.79 ± 1.20	1.212 ± 0.016
Bayesian Optimization	5.26 ± 0.40	6.98 ± 2.60	1.960 ± 0.288
BED	31.58 ± 0.00	39.47 ± 1.50	0.249 ± 0.015
LLM-only	0.00 ± 0.00	0.00 ± 0.00	0.764 ± 0.189
Code-assisted LLM	0.00 ± 0.00	0.00 ± 0.00	0.824 ± 0.183
LLM-AutoSciLab (Ours)	**35.09 ± 0.80**	**50.88 ± 2.60**	**0.189 ± 0.004**
Method	**Edge F1**	**Exact Graph**	**Sign**
	ActiveSciBench-GRN		
GENIE3	38.60 ± 0.00	0.00 ± 0.00	79.07 ± 1.60
GIES	56.27 ± 1.80	6.67 ± 1.33	85.78 ± 2.30
NOTEARS	27.60 ± 1.20	2.22 ± 0.00	58.15 ± 1.30
Random sampling	35.56 ± 3.80	2.22 ± 0.00	84.44 ± 2.00
Uncertainty sampling	50.10 ± 1.40	4.44 ± 0.00	88.15 ± 2.00
LLM-only	50.41 ± 3.50	0.00 ± 0.00	79.44 ± 0.60
Code-assisted LLM	54.67 ± 4.70	0.00 ± 0.00	85.98 ± 0.90
LLM-AutoSciLab (Ours)	**72.49 ± 1.30**	**31.11 ± 1.33**	**98.15 ± 0.30**

To assess statistical significance, we repeated each benchmark configuration across three seeds and report the mean and standard deviation of the seed-level aggregate scores. The mean results are presented in Table 3. For NewtonBench and ActiveSciBench-Chem, we report symbolic accuracy (SA), exact accuracy, and RMSLE. For ActiveSciBench-GRN, we report edge F_1, exact graph accuracy, and sign accuracy. Overall, the variance across benchmarks is modest. The strongest methods, including LLM-AutoSciLab, remain stable across seeds, while weaker LLM-only or unguided baselines show somewhat larger variability, particularly on harder graph-recovery settings. Standard deviations of ± 0.00 indicate that the seed-level aggregate scores are identical at the displayed precision as expected for deterministic baselines or methods run under fixed random seeds.

B Benchmark Details

B.1 ActiveSciBench-Chem

ActiveSciBench-Chem is an active enzyme kinetics benchmark for mechanism recovery under finite experimental budgets. Rather than receiving a fixed dataset, the learner adaptively selects biochemical assay conditions and observes reaction rates to infer hidden kinetic mechanisms and parameters. This setup reflects the challenge of scientific discovery, where multiple mechanisms can explain limited observations and targeted experiments are required to distinguish competing hypotheses.

$$\mathbf{x} = (C_A, C_I, C_B, C_P, \mathrm{Enz}, T, \mathrm{pH}),$$

covering substrate, inhibitor, secondary substrate, product concentration, enzyme loading, temperature, and pH. While every task exposes the same interface, each mechanism depends on only a subset of variables, requiring the learner to identify both the governing law and the relevant dimensions. Dependence on different variables corresponds to distinct mechanistic behaviors such as inhibition, bisubstrate reactions, product feedback, or environmental modulation. ActiveSciBench-Chem contains **57 curated tasks** organized into easy, medium, and hard tiers. Easy tasks correspond to standard kinetic families, medium tasks introduce structured compositions, and hard tasks include weaker identifiability and nonstandard behaviors such as cooperative or allosteric kinetics. The ActiveSciBench-Chem benchmark evaluates both active experimentation and symbolic scientific reasoning and is organized around **nine canonical base families**:

- Michaelis–Menten saturation
- Competitive inhibition
- Product inhibition
- Arrhenius temperature dependence
- Ping-pong bisubstrate kinetics
- Uncompetitive inhibition
- Substrate inhibition
- Hill cooperativity
- Noncompetitive inhibition

Beyond the nine base families, ActiveSciBench-Chem includes **structured composite mechanisms** that combine substrate-response kinetics with modifiers such as inhibition, temperature dependence, or feedback. Examples include Michaelis–Menten with competitive inhibition and Arrhenius modulation, ping-pong bisubstrate kinetics with noncompetitive inhibition, and Hill cooperativity with product feedback. These composites move beyond isolated textbook mechanisms and require the learner to distinguish competing mechanistic explanations under a shared assay interface. ActiveSciBench-Chem also includes a targeted set of **extended and nonstandard mechanism families** beyond the core base library:

- Ordered sequential bisubstrate kinetics
- Allosteric activation
- Anti-cooperative Hill behavior
- Fractal or anomalous kinetics
- Mixed inhibition
- Cooperative inhibition
- Monotonic pH dependence
- Metal-ion activation
- Product activation / autocatalytic feedback
- Dual inhibition by inhibitor and product

Taken together, the 57 ActiveSciBench-Chem tasks span textbook kinetic families, structured compositions, and targeted extended mechanisms beyond the standard library. This makes ActiveSciBench-Chem both interpretable at the mechanism level and sufficiently rich to require active experimentation, relevant-variable identification, and mechanistic discrimination rather than simple equation retrieval. ActiveSciBench-Chem is a simulator-based oracle benchmark designed to isolate active kinetic mechanism discovery under controlled, reproducible conditions. Observations are generated from mechanistically specified kinetic families with hidden parameters and benchmark-defined noise, rather than physical wet-lab experiments. While it does not capture the full complexity of laboratory biochemistry, such as assay failures, batch effects, protocol variability, or experimental cost, it provides a clean, budget-controlled setting for evaluating closed-loop mechanistic recovery.

B.2 ActiveSciBench-GRN

ActiveSciBench-GRN is an online gene perturbation benchmark for active causal graph discovery in gene regulation. Unlike ActiveSciBench-Chem, which focuses on recovering biochemical rate laws, ActiveSciBench-GRN requires the learner to infer hidden causal structure and nonlinear dynamics from interventional data. Each task simulates a hidden regulatory system with unknown graph structure, edge signs, and nonlinear dynamics. The learner performs discrete interventions, such as gene knock-up or knock-down experiments, and observes downstream expression changes, while

the underlying graph remains unobserved. Different motifs can produce similar responses under limited interventions, and hidden parameters can make identical motifs appear qualitatively different, requiring the learner to jointly infer topology, sign, and dynamics from sparse experimental data. ActiveSciBench-GRN is built from canonical regulatory motifs spanning increasing structural complexity, including activation, repression, feedback, and switching behavior. The benchmark contains five core motif families, three topological variants, and three difficulty levels, yielding 45 tasks per random seed. These motifs are standard systems biology primitives that provide a compact yet meaningful testbed for mechanistic graph discovery. The five ActiveSciBench-GRN motif families are:

- **Activation chain.** A layered activation cascade from signal to intermediate regulators to the reporter.
- **Coherent feedforward loop.** A motif in which the input acts through both a direct and a mediated activating branch.
- **Incoherent feedforward loop.** A motif in which activation and repression act along competing paths, producing adaptation-like or pulse-like behavior.
- **Negative-feedback circuit.** A self-limiting repression architecture in which downstream activation induces a repressive branch.
- **Toggle-switch or bistable decision circuit.** A mutually repressive switching architecture used to model bistability, state selection, and cellular decision making.

Each motif family is instantiated across three topological variants and three difficulty levels. The topological variants preserve the motif identity while changing the precise wiring structure, whereas the difficulty levels correspond to increasingly nonlinear and feedback-sensitive parameter regimes. Easy tasks exhibit near-linear responses that reveal the graph relatively clearly, while medium and hard tasks introduce saturation, switching behavior, and bistability that obscure weak or mediated dependencies. As a result, ActiveSciBench-GRN is not simply a passive graph estimation problem, but a budgeted intervention design task in which the learner must select perturbations that best distinguish competing regulatory mechanisms. ActiveSciBench-GRN is a simulator-based oracle benchmark designed to isolate intervention-driven regulatory discovery under controlled, reproducible conditions. Responses are generated from motif-specific nonlinear dynamics with hidden parameters and benchmark-defined noise rather than physical biological experiments. Although it does not capture the full complexity of real perturbation biology, including failed interventions, cell-state heterogeneity, batch effects, off-target effects, or experimental cost variability, it provides a clean, budget-controlled testbed for evaluating closed-loop mechanistic graph recovery.

B.3 NewtonBench

NewtonBench provides the physics component of our benchmark suite. In contrast to ActiveSciBench-Chem and ActiveSciBench-GRN, it isolates active symbolic law discovery in a setting where the relevant variables for each task are already known. In our experiments, one oracle call evaluates the hidden physical law at a chosen assignment of task-specific input variables, and the learner must recover the symbolic law under a finite query budget. Because NewtonBench is already introduced as a standalone benchmark in [52], we refer readers there for the full benchmark construction, counterfactual law-generation procedure, and task catalog. Table 7 summarizes the benchmark suite we use for evaluation in this work.

C Metrics

We evaluate all three benchmarks under a unified objective of recovering the true scientific mechanism. For NewtonBench and ActiveSciBench-Chem, we report three metrics. First, we measure predictive fidelity using the root mean squared logarithmic error (RMSLE),

$$\mathrm{RMSLE}(\hat{f}, f) = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(\log(1+\hat{y}_i) - \log(1+y_i)\right)^2}.$$

RMSLE is appropriate because target values in these domains can span multiple orders of magnitude. Second, we report numerical exact accuracy,

$$\mathrm{ExAcc}(\hat{f}) = \mathbf{1}\Big[\mathrm{RMSLE}(\hat{f}, f) < 0.01\Big],$$

Table 7: **Benchmark summary.** NewtonBench, ActiveSciBench-Chem, and ActiveSciBench-GRN differ in scientific setting, discovery target, interface, and benchmark families.

Attribute	Description
NewtonBench	
Scientific setting	Active physics law discovery in known-variable domains.
Discovery target	Symbolic physical law.
Interface	Task-specific physical variables; one oracle call evaluates the hidden law at a chosen assignment of inputs.
Setting used in this work	`vanilla_equation`.
Core domains	Newtonian gravitation; Coulomb force; magnetic force; Fourier heat conduction; Snell's law; radioactive decay; underdamped harmonic motion; Malus's law; speed of sound; Hooke's law; Bose–Einstein distribution; heat transfer.
Domain count	12.
Role in benchmark suite	Isolates active symbolic law discovery when the relevant variables are already known.
ActiveSciBench-Chem	
Scientific setting	Active enzyme-kinetics discovery under a shared biochemical assay interface.
Discovery target	Symbolic kinetic law with hidden relevant-variable structure.
Interface	Shared 7D biochemical assay interface: $(C_A, C_I, C_B, C_P, \mathrm{Enz}, T, \mathrm{pH})$.
Primary observation	Initial rate r_0 together with auxiliary mass-balance observables.
Base families	Michaelis–Menten saturation; competitive inhibition; product inhibition; Arrhenius temperature dependence; ping-pong bisubstrate kinetics; uncompetitive inhibition; substrate inhibition; Hill cooperativity; noncompetitive inhibition.
Composite mechanisms	Structured compositions of the base families, combining substrate-response families with modifiers such as inhibition, temperature dependence, feedback, and bisubstrate effects. Representative examples include MM with competitive inhibition and Arrhenius modulation, ping-pong with noncompetitive inhibition, Hill cooperativity with product modulation, and substrate inhibition with Arrhenius modulation.
Extended families	Ordered sequential bisubstrate kinetics; allosteric activation; anti-cooperative Hill behavior; fractal / anomalous kinetics; mixed inhibition; cooperative inhibition; monotonic pH dependence; metal-ion activation; product activation / autocatalytic feedback; dual inhibition by inhibitor and product.
Domain count	57 curated domain classes.
Role in benchmark suite	Adds hidden relevant-variable discovery and mechanistically grounded biochemical ambiguity beyond known-variable symbolic regression.
ActiveSciBench-GRN	
Scientific setting	Intervention-driven regulatory discovery from perturbation-response experiments.
Discovery target	Signed regulatory graph / motif together with effective nonlinear regulatory structure.
Interface	Shared perturbation interface with discrete interventions over hidden regulatory systems.
Primary observation	Downstream expression changes following intervention.
Motif families	Activation chain; coherent feedforward loop; incoherent feedforward loop; negative-feedback circuit; toggle-switch / bistable decision circuit.
Topological variants	Three versions per motif family.
Difficulty levels	Easy, medium, and hard parameter regimes, corresponding to progressively stronger nonlinearity and more difficult motif disambiguation.
Family count	5 core motif families.
Role in benchmark suite	Adds intervention-driven signed graph discovery and causal-mechanism ambiguity beyond equation recovery alone.

which follows the exact-recovery criterion of NewtonBench [52]. Third, we report symbolic accuracy (SA), which measures whether the recovered law matches the ground-truth mechanism up to algebraic rewriting, fitted constants, and variable renaming. Symbolic accuracy is stricter than numerical exactness, since a numerically accurate approximation need not recover the correct mechanistic form. For ActiveSciBench-GRN, the target is graph recovery rather than scalar law recovery. We therefore report edge-level precision, recall, and F1, sign accuracy for activation versus repression, exact graph accuracy, and motif accuracy. These metrics distinguish partial structural recovery from full mechanistic recovery. For NewtonBench and ActiveSciBench-Chem, symbolic accuracy is evaluated with an LLM judge that determines whether the predicted hypothesis is equivalent to the ground-truth expression up to constant parameter values. In this evaluation, the ground-truth law is presented as expression A, and the candidate hypothesis B may be represented either as an executable program or as a symbolic expression. As illustrated, the judge is prompted as follows:

```
Question: Given the ground truth mathematical expression A and the
hypothesis B, determine if there exist any constant parameter values that
would make the hypothesis equivalent to the given ground truth expression.
Let's think step by step. Explain your reasoning and
then provide the final answer as:
{
  "reasoning": "Brief step-by-step analysis",
  "answer": "Yes/No"
}
```

A prediction is counted as symbolically correct if the judge returns "Yes." This metric is stricter than numerical exactness, since a numerically accurate approximation need not recover the same underlying symbolic form.

D ActiveSciBench Design and Memorization

A key concern for LLM-based scientific discovery is whether strong performance reflects genuine mechanistic recovery or simple recall of textbook systems. Our benchmarks are designed to make retrieval alone insufficient. Each run begins without observations, requiring the learner to actively select experiments under a limited oracle budget and infer the hidden mechanism from the resulting responses. The instantiated system, not just the family label, is hidden throughout. In NewtonBench, the learner observes the natural variables but not the hidden governing law. In ActiveSciBench-Chem, all tasks share a common seven-variable assay interface while the relevant variables, mechanism class, and parameterization remain hidden. In ActiveSciBench-GRN, the learner must infer a hidden signed regulatory mechanism from perturbation responses rather than directly observing the graph. At the same time, realistic scientific semantics are preserved through variables such as substrate, inhibitor, temperature, and perturbation target, ensuring the evaluation measures scientific reasoning rather than anonymized symbol matching. This can also be quantified through conservative lower bounds on the hidden task space. In ActiveSciBench-Chem, the 57 mechanism classes contain between 2 and 8 hidden continuous parameters (median 5). Even under a coarse discretization of only 100 possible values per parameter, this yields more than 6×10^{16} possible hidden instantiations. ActiveSciBench-GRN is broader still: the target is a signed regulatory graph over $signal, A, B, C, R$ with 20 possible directed non-self edges, each absent, activating, or repressing, yielding $3^{20} \approx 3.5 \times 10^9$ unconstrained signed graphs. Even with a sparsity cap of six edges, this leaves more than 3 million candidate graphs before accounting for hidden continuous dynamics. These values are conservative lower bounds, but they illustrate that the benchmarks cannot be reduced to retrieving a small library of familiar mechanisms.

E Implementation Details

E.1 Baselines

All methods use the same oracle, task instance, difficulty, law version, and total experiment budget as LLM-AutoSciLab. For the **LLM-only** and **Code-assisted LLM** conditions, we follow the same definitions and prompting/tool-use setups as in NewtonBench and refer readers to [52] for those

implementation details. Below, we summarize the remaining comparison methods used in our benchmark suite.

Random+PySR. This method uses a non-adaptive experimental design followed by symbolic regression. It serves as the non-adaptive floor.

BO+PySR. This method uses Gaussian-process Bayesian optimization without LLM guidance.

BED+PySR. This method performs Bayesian experimental design over a fixed hand-specified mechanism library. At each step, each candidate mechanism is fit to the current observations by nonlinear least squares in log-space, candidate experiments are sampled from the admissible bounds, and the next experiment is chosen to maximize the disagreement between the fitted mechanisms. After the budget is exhausted, the best-fitting library member is retained, and PySR is run as a final symbolic refinement stage. This uses the same general style of mechanistic library reasoning as the LLM pipeline, but with a fixed, predefined family set rather than dynamically generated hypotheses.

GENIE3. We evaluate GENIE3 on fixed ActiveSciBench-GRN datasets collected under the same task budget and use the standard `GENIE3` implementation [25], with a lightweight post-processing step to convert the output into the signed-graph format required by ActiveSciBench-GRN evaluation.

GIES. We evaluate GIES on the same fixed ActiveSciBench-GRN datasets using the standard `pcalg` implementation [23], with intervention labels corresponding to the benchmark perturbation environments and the same signed-graph conversion.

NOTEARS. We evaluate the linear NOTEARS method on the same fixed ActiveSciBench-GRN datasets using the reference `notears` implementation [53], followed by the same signed-graph conversion used for benchmark evaluation.

LLM-only and Code Assisted LLM. We follow the implementation of these baselines as presented in NewtonBench [52].

LLM-SR+EIG and SR-Scientist+EIG. We implement LLM-SR+EIG and SR-Scientist+EIG using the official LLM-SR[2] and SR-Scientist[3] repositories. Since both methods normally operate on a fixed dataset, we use them as equation-fitting backends within an active experimental loop. At each step, the method is fitted to all observations collected so far and its valid, distinct equations are retained as candidate models. We assign an initially uniform prior over these candidates and compute their posterior probabilities from how well they explain the observed data under a Gaussian observation model. For each possible experiment, we estimate the expected information gain as the expected reduction in entropy over the candidate models after observing the experiment's possible outcome. The expectation is approximated by sampling outcomes from the candidates' predictive mixture. We execute the experiment with the largest estimated information gain, add its outcome to the dataset, and refit the symbolic-regression method before selecting the next experiment. Each task is limited to 100 oracle experiments in total, including the initial space-filling observations; fitting equations and evaluating hypothetical experiments do not consume this budget. After the budget is exhausted, we fit the underlying method once more to the complete dataset and evaluate its highest-posterior equation using symbolic accuracy, exact accuracy, and RMSLE.

E.2 LLM-AutoSciLab

Our implementation of LLM-AutoSciLab instantiates the closed-loop discovery framework in a hypothesis-driven setting with adaptive ensembling. At each stage, the system maintains a set of candidate mechanisms, proposes informative experiments conditioned on disagreement among those candidates, updates the observation set with oracle feedback, and refines the candidate laws through symbolic fitting after every acquired observation. The implementation uses a two-model architecture in which a smaller local model generates a diverse hypothesis set, while the main model synthesizes these hypotheses with accumulated evidence to guide subsequent experimentation. Symbolic refinement results are summarized in a structured memory representation and re-injected into later reasoning steps.

[2] `https://github.com/deep-symbolic-mathematics/LLM-SR`
[3] `https://github.com/GAIR-NLP/SR-Scientist`

Prompting and Hypothesis Generation. The hypothesis-generation step receives the current goal, domain description, parameter names, Python function signature, experiment history as a text table, the current working hypothesis, the remaining budget, the current phase, the best symbolic equation found so far (if available), and the structured memory summary. At this stage, the model is asked to propose one primary hypothesis together with multiple alternate hypotheses, but *not* search regions. All hypotheses are returned as pure Python expressions using the exact oracle variable names and symbolic free constants such as `C0`, `C1`, `alpha`, and `beta`. In our configuration, the smaller ensemble model is sampled in parallel with base ensemble size $K = 5$, and each call produces a structured hypothesis output from which the primary candidate is retained for ensemble construction.

Algorithm 2 Adaptive Ensemble

Require: Prompt context S_t, base ensemble size K, cap $K_{\max}$, entropy threshold τ_H

$\mathcal{C} \leftarrow \emptyset,\ H_{\text{prev}} \leftarrow \varnothing$
while $|\mathcal{C}| < K_{\max}$ **do**
  $b \leftarrow \min(K, K_{\max} - |\mathcal{C}|)$
  $\mathcal{B} \leftarrow \texttt{SampleHypotheses}(\pi_\theta^{\text{small}}, S_t, b)$
  $\mathcal{C} \leftarrow \mathcal{C} \cup \texttt{FilterValid}(\mathcal{B})$
  if $|\mathcal{C}| < K$ **then**
    continue
  end if
  # Cluster $\mathcal{C}$ by structural skeleton and compute entropy H
  if $H_{\text{prev}} \neq \varnothing$ **then**
    if $|H - H_{\text{prev}}| < \tau_H$ **then**
      break
    end if
  end if
  $H_{\text{prev}} \leftarrow H$
end while
return $\texttt{BuildDistribution}(\mathcal{C})$

Adaptive Ensemble Construction. Raw ensemble hypotheses are filtered before use: expressions that cannot be executed on the observed data, produce predominantly non-finite predictions, or are clearly nonsensical under the current dataset are discarded. The remaining hypotheses are then clustered by structural skeleton, obtained by canonicalizing constants while preserving functional form. LLM-AutoSciLab uses adaptive ensemble growth in which hypotheses are sampled in batches of size K, reclustered after each batch, and the Shannon entropy of the structural cluster distribution is recomputed. Sampling stops when the entropy change between successive batches falls below 0.1, or when the hard cap of $K_{\max} = 20$ samples is reached. The resulting hypothesis distribution stores the raw hypotheses, structurally unique representatives, cluster assignments, the majority-cluster agreement score, and a synthesis summary that is passed to the main model.

Hypothesis-Conditioned Acquisition. The main model receives a second prompt containing the full experiment history, the current working hypothesis, the structured memory summary, and a compact summary of the current hypothesis set, including the number of sampled hypotheses, the number of unique structures, and representative candidates from each structural cluster. It returns an updated primary hypothesis, alternate hypotheses, and a set of search regions. In parallel, LLM-AutoSciLab computes a falsification-oriented disagreement score directly from the current hypothesis set. For a candidate point $\mathbf{x}$, let $\hat{f}_1(\mathbf{x}), \ldots, \hat{f}_K(\mathbf{x})$ denote the predictions induced by the current candidate mechanisms. We define the disagreement score as

$$\Delta(\mathbf{x}) = \mathrm{Std}\Big(\log_{10} \hat{f}_1(\mathbf{x}), \ldots, \log_{10} \hat{f}_K(\mathbf{x})\Big)\,.$$

For ActiveSciBench-GRN, disagreement is computed over fitted graph hypotheses through their predicted intervention responses. If graph hypothesis g_k predicts a post-intervention response vector $\hat{\mathbf{y}}_k(a) \in \mathbb{R}^d_{>0}$ for intervention a, we define

$$\delta_{\mathrm{GRN}}(a) = \frac{1}{d} \sum_{\ell=1}^{d} \mathrm{Var}_{k=1,\ldots,K}\Big[\log\big(\hat{y}_{k,\ell}(a) + \varepsilon\big)\Big].$$

This means that edge and sign disagreements matter only through their falsifiable intervention consequences. The role of the LLM at this stage is to propose search regions expected to be informative for separating competing mechanisms; the final experiment points are then selected by the active-learning layer within those regions. When the system is in the low-confidence regime, candidate points are sampled from the proposed bounds, scored by $\Delta(\mathbf{x})$, and a small diverse subset is chosen for oracle evaluation. This yields a hypothesis-conditioned acquisition rule that preferentially queries regions where competing mechanisms make sharply different predictions.

Hypothesis Refinement. Mechanism refinement is performed on accumulated observations using a domain-specific refinement backend after every acquired observation. Before the backend is invoked,

the loop extracts the variables implicated by the current hypothesis set and uses them to focus the refinement search, optionally reintroducing variables whose residual behavior suggests missing structure. In equation-discovery settings, refinement combines direct fitting of candidate structural families with numerical parameter optimization and symbolic search over the accumulated observations. In graph-discovery settings, refinement updates the candidate signed regulatory structures and their associated dynamical parameters using the observed perturbation responses. The refined candidates are then pooled for later arbitration and memory updates.

Bootstrap Confidence. Confidence is computed in bootstrap mode and is used to control the acquisition regime rather than to terminate the run. After fitting a candidate mechanism, the domain-specific refinement backend is rerun on bootstrap resamples of the training split, and the resulting models are evaluated on a held-out validation split. Let $\hat{y}^{(b)}$ denote the prediction vector from bootstrap fit b. We compute bootstrap confidence from the mean coefficient of variation across validation predictions,

$$\mathrm{conf}_{\mathrm{boot}} = 1 - \frac{1}{N}\sum_{i=1}^{N} \frac{\mathrm{std}_b\left(\hat{y}_i^{(b)}\right)}{\left|\mathrm{mean}_b\left(\hat{y}_i^{(b)}\right)\right| + \varepsilon},$$

clipped to $[0, 1]$. In our setup, this confidence determines whether acquisition emphasizes hypothesis disambiguation or parameter refinement. When bootstrap confidence remains below the gating threshold, the system stays in a disagreement-driven regime and selects experiments using hypothesis-conditioned acquisition to separate competing mechanisms. Once confidence exceeds the threshold (0.9 in our configuration), the system switches to a refinement regime, where acquisition is driven by uncertainty sampling to reduce residual uncertainty within the current high-confidence mechanism class.

Memory and Final Selection. The memory injected into later prompts is a structured summary rather than a raw conversation transcript. It can include the full symbolic-regression history, the current best equation and its fit statistics, bootstrap confidence, computed RMSLE based on experiment data and proposed hypotheses, validated hypotheses that survived discriminating experiments, negative evidence for falsified forms, and a hypothesis scoreboard that tracks validated, failed, and uncertain structures. At budget exhaustion, LLM-AutoSciLab performs a final symbolic fitting pass using the full symbolic-regression budget, reuses the final variable filter and hypothesis-family pool, and constructs a final candidate set from domain-specific optimization candidates, direct skeleton fits, and validated hypothesis survivors. The main model then arbitrates among these final candidates based on scientific plausibility and consistency with the observed data, rather than selecting solely on training R^2. If a mid-run candidate achieved a clearly better held-out observed-data validation score, computed without access to the ground-truth mechanism, that candidate is preserved as the final equation.

E.3 Experimental Protocol

Unless otherwise noted, all experiments use deterministic benchmark instances with zero oracle noise and matched task manifests. NewtonBench evaluates 12 domains across 3 difficulty levels, 3 law versions, and 3 seeds (324 runs total), with representative budget studies using 96 tasks. For ActiveSciBench-Chem, representative manifests contain 36 tasks for ablation and analysis, with standard budgets $B \in \{20, 40, 60, 80, 100\}$ and the entire dataset for main results. For ActiveSciBench-GRN, representative manifests contain 36 tasks for budget studies and 18 for noise studies, using $B \in \{10, 20, 50\}$ with fixed-budget comparisons at $B = 20$ on the entire dataset for main results. NewtonBench fixed-budget comparisons also use $B = 20$, with extended studies reported up to $B = 100$. Evaluation is performed on held-out oracle outputs. Symbolic regression tasks report RMSLE-based recovery together with exact and symbolic recovery, while ActiveSciBench-GRN reports edge F_1, exact graph accuracy, and sign accuracy against the hidden graph. Unless a row is explicitly labeled Claude Sonnet 4.6, LLM-based experiments use `GPT-4o-mini` as the primary reasoning model. Claude Sonnet 4.6 was accessed through Amazon Bedrock on AWS. LLM-AutoSciLab uses `Qwen/Qwen2.5-7B-Instruct` for adaptive ensembling via local vLLM. Models are used without task-specific fine-tuning, and ensemble sampling uses a temperature of 1.0 for structural diversity. Symbolic regression refinement uses PySR with 800 iterations plus direct fitting of candidate mechanism skeletons when available, while ActiveSciBench-GRN uses signed-graph fitting with BFGS optimization. Bootstrap confidence from held-out validation fits is used only for acquisition-mode switching.

E.4 Prompt Templates

Hypothesis Generation (Symbolic Regression)

```
GOAL: {goal}

DOMAIN: {domain}
PARAMETERS: {param_names}
{param_description}
OBJECTIVE TYPE: {objective_type}
OBJECTIVE DIRECTION: {objective_direction}
{optional_objective_profile}

FUNCTION SIGNATURE:
{function_signature}

DATA ({budget_pct}% budget used, {budget_remaining} experiments left):
{data_table}

Best symbolic equation so far:
{best_equation_fit}

{memory_str}
Current hypotheses: {current_hypothesis}
CURRENT PHASE: {current_phase}

TASK:
Generate one primary and 2-6 alternate EQUATION hypotheses that remain
plausible under current data.
Do NOT propose search regions in this step.
Focus only on plausible competing hypotheses and concise reasoning for
ambiguity.
```

Search-Region Proposal (Symbolic Regression)

```
GOAL: {goal}

DOMAIN: {domain}
PARAMETERS: {param_names}
{param_description}
OBJECTIVE TYPE: {objective_type}
OBJECTIVE DIRECTION: {objective_direction}
{optional_objective_profile}

DISCOVERED LAW FUNCTION SIGNATURE (must match exactly):
{function_signature}

EXPERIMENTAL DATA ({budget_pct}% of budget used, {budget_remaining}
experiments remaining):
{data_table}

{best_equation_if_present}
{hypothesis_requirement_if_present}
Current best hypothesis: {current_hypothesis}
{memory_str}
{phase_instruction}
{discrimination_hints}

Based on this data, propose the most informative parameter regions to
explore next. Think carefully about what the data represents and what
experiments would most help discover the governing equation via disambiguating the competing hypotheses set.

search_regions must be a list of objects with this exact structure:
{"bounds": {"p1": [lo, hi], "p2": [lo, hi], ...},
 "n_experiments": 4,
 "priority": "high",
 "rationale": "why"}
```

Hypothesis-to-Executable Structure (Symbolic Regression)

```
GOAL: {goal}

DOMAIN: {domain}
PARAMETERS: {params_str}

REQUIRED FUNCTION SIGNATURE (use EXACTLY this):
{function_signature}

EXPERIMENTAL DATA (raw values and [log10 values]):
{data_table}

CURRENT HYPOTHESIS: {current_hypothesis}
{previous_attempts_if_any}

Read the log10 columns to estimate exponents:
\Delta log10(measurement) / \Delta log10(param) \approx exponent.
Then propose a Python function 'discovered_law' that fits this
structural form.

RULES:
- Use free constants C0, C1, alpha, beta, ... - they will be fitted
  numerically.
- Use ONLY standard Python math (no imports). Powers: use ** not pow().
- Use the EXACT parameter names from the FUNCTION SIGNATURE above.
- Constants must appear in the return expression.
```

Hypothesis Generation and Region Proposal (Graph Recovery)

```
GOAL:
{goal}

DOMAIN:
{oracle.param_description}

CURRENT BEST HYPOTHESIS:
{current_hypothesis}

RECENT DATA:
{store_to_table}

HYPOTHESIS FIT SUMMARY:
{lineage_summary}

MEMORY SUMMARY:
{memory_str}

CONFIDENCE SUMMARY:
{confidence_str}

ENSEMBLE GRAPH DISTRIBUTION:
{ensemble_summary}

{optional_diversity_requirement_block}

BUDGET REMAINING: {budget_remaining}
MAX EXPERIMENTS THIS ITERATION: {max_experiments_per_iter}

Return:
- 2-5 natural-language mechanism hypotheses with stable IDs like h1,
  h2, h3
- a primary_hypothesis_id naming the current best one
- 1-4 search regions for the next experiments
- confidence and done flag
```

Single-Hypothesis Sampling (Graph Recovery)

```
GOAL:
{goal}

DOMAIN:
{oracle.param_description}

CURRENT BEST HYPOTHESIS:
{current_hypothesis}

RECENT DATA:
{store_to_table}

HYPOTHESIS FIT SUMMARY:
{lineage_summary}

MEMORY SUMMARY:
{memory_str}

CONFIDENCE SUMMARY:
{confidence_str}

BUDGET REMAINING: {budget_remaining}

Return exactly one plausible natural-language mechanism hypothesis,
with a short rationale and confidence.
Do not output graph edges or search regions.
```

Hypothesis-to-Structure Translation (Graph Recovery)

```
Translate this single natural-language hypothesis into one signed graph.

hypothesis_id: {hypothesis_id}
text: {hypothesis_text}

Fill this schema exactly:
{
  "translation": {
    "hypothesis_id": "{hypothesis_id}",
    "rationale": "why this graph matches the hypothesis",
    "assumptions": ["optional assumption"],
    "edges": [
      {"src": "signal", "dst": "A", "sign":1},
      {"src": "A", "dst": "C", "sign":1}
    ]
  }
}

Requirements:
- dst must be one of {graphs_nodes}.
- The graph must contain a directed path from signal to {final_node}.
- If the hypothesis does not mention {final_node} explicitly, add the minimal
  faithful chain needed so the signal reaches {final_node}.
```

NeurIPS Paper Checklist

1. **Claims**
   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?
   Answer: [Yes]
   Justification: The results in Section 5 and the analysis in Section 6 support the claims made in the abstract and introduction.
   Guidelines:

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

2. **Limitations**
   Question: Does the paper discuss the limitations of the work performed by the authors?
   Answer: [Yes]
   Justification: Our work evaluates with simulator-based oracles with oracle noise set to zero by default; a noise sensitivity analysis is provided in Appendix A.1. Some implicit assumptions underlie this evaluation that may not hold in physical settings: conditional independence (successive queries are independent), and refinement convergence (800-iteration PySR and BFGS budgets are sufficient for the mechanisms evaluated). Physical wet-lab deployment introduces additional challenges such as reagent variability, calibration drift, and heterogeneous experiment cost, but these are outside the current scope.
   Guidelines:

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

3. **Theory assumptions and proofs**

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?
Answer: [N/A]
Justification: No theoretical proofs are included in the paper.
Guidelines:

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

4. **Experimental result reproducibility**
   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?
   Answer: [Yes]
   Justification: We provide pseudocode and data in the main paper and appendix, together with code and reproduction instructions at `https://github.com/scientific-discovery/LLM-AutoSciLab`.
   Guidelines:

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

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

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?
Answer: [Yes]
Justification: We provide open-source data, code, and execution instructions at `https://github.com/scientific-discovery/LLM-AutoSciLab`.
Guidelines:

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

6. **Experimental setting/details**
Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?
Answer: [Yes]
Justification: The section 5 of main paper and appendix lists all the details necessary to understand the results.
Guidelines:

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

7. **Experiment statistical significance**
Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?
Answer: [Yes]
Justification: We report standard deviation across multiple seeds.
Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.

- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**
   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?
   Answer: [Yes]
   Justification: We use 4 × Quadro RTX 8000 48GB GPUs to host open models and recommend at least 16GB RAM for domain-specific optimization with PySR and BFGS. GPT-4o-mini is accessed through the OpenAI API, while Claude Sonnet 4.6 is accessed through Amazon Bedrock on AWS.
   Guidelines:
   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).
9. **Code of ethics**
   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?
   Answer: [Yes]
   Justification: The work complies with NeurIPS ethical standards.
   Guidelines:
   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).
10. **Broader impacts**
    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?
    Answer: [Yes]
    Justification: The paper contains an impact statement discussing the societal impacts.
    Guidelines:
    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out

that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.

- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

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

    Answer: [N/A]

    Justification: The work does not release any information requiring safeguards to the best of our knowledge.

    Guidelines:

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

12. **Licenses for existing assets**

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

    Answer: [Yes]

    Justification: External assets are credited in the paper and repository. PySR and Qwen2.5-7B-Instruct use Apache-2.0; NewtonBench uses MIT; GENIE3 and pcalg/GIES use GPL (version 2 or later); and NOTEARS uses Apache-2.0. Closed API models are used under the applicable OpenAI and AWS service terms. Our released code is licensed under MIT, with third-party components retaining their original licenses.

    Guidelines:

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

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?
Answer: [Yes]
Justification: The released benchmark data and MIT-licensed code are documented with setup and execution instructions at `https://github.com/scientific-discovery/LLM-AutoSciLab`.
Guidelines:

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

14. **Crowdsourcing and research with human subjects**
Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?
Answer: [N/A]
Justification: No crowdsourcing was involved.
Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**
Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?
Answer: [N/A]
Justification: No human subjects were involved
Guidelines:

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

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

Justification: LLMs are a core component of LLM-AutoSciLab, where they generate and synthesize candidate hypotheses and propose informative experiment regions; we also evaluate LLM-based baselines.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.