Automated Capability Evaluation of Foundation Models

Anonymous Author(s)
Affiliation
Address
email

Abstract

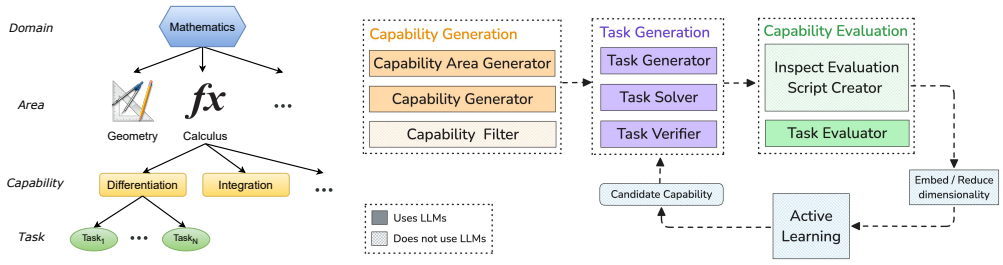
Current evaluation frameworks for foundation models rely on fixed, manually curated benchmarks, limiting coverage of model capabilities. We propose Active learning for Capability Evaluation, a scalable framework for automated fine-grained evaluation. Our framework leverages language models to decompose domains into semantically meaningful capabilities and generate diverse tasks, reducing human effort. It models a subject model's performance as a capability function over a latent semantic space and applies active learning to prioritize the most informative evaluations. This adaptive strategy enables cost-efficient discovery of strengths, weaknesses, and failure modes that static benchmarks may overlook. Results show that this evaluation yields a more complete picture of model capabilities.

1 Introduction

As foundation models expand in scale, rigorous evaluation of their capabilities is crucial. Evaluations guide model selection, inform development, and ensure safety in high-stakes domains such as cybersecurity and healthcare. Most of the current evaluations rely on static, human-curated benchmarks [1, 2, 3, 4, 5, 6, 7, 8, 9]. While valuable, such benchmarks lag behind the pace of model development. In addition they overlook fine-grained skills, and are costly to create and maintain. Large Language Models (LLM) enable a new paradigm: automated generation of semantically meaningful capabilities and diverse tasks. However, scalability remains a bottleneck. A single domain may contain thousands of capabilities, each requiring extensive tasks, making exhaustive evaluation, especially of commercial models, expensive. We address this with a data-efficient approach based on active learning. Instead of exhaustive coverage, we prioritize the most informative capabilities by reducing uncertainty. A key notion here is the *capability function*, which maps latent capability representations to a model performance score. Modeling this function allows interpolation across related capabilities and principled selection of what to evaluate next.

We introduce **A**ctive Learning for **C**apability **E**valuation (ACE), a framework for scalable, automated, and fine-grained evaluation. ACE (1) uses LLMs to decompose domains into structured capabilities and generate task sets, and (2) actively evaluates models by learning the capability function in latent space and adaptively selecting informative capabilities. The codebase is available at https://anonymous.4open.science/r/ace-7EAF. Our contributions are: **(i)** A general framework (ACE) combining LLM-based capability decomposition and task generation with active learning for scalable evaluation. **(ii)** Extensive experiments in Mathematics with 78 capabilities and 8,500+ tasks, evaluating multiple open- and closed-source models and uncovering differences invisible to aggregate metrics. **(iii)** Empirical evidence that the latent space constructed from pretrained text encoders preserves semantic structure, enabling effective approximation of the capability function. **(iv)** Validation of automatically generated tasks and verification outputs via manual inspection, showing strong agreement with human labels.

Submitted to 39th Conference on Neural Information Processing Systems (NeurIPS 2025). Do not distribute.



(a) (b)

Figure 1: (a) The hierarchy of capabilities (b) The ACE pipeline

2 Active Learning for Capability Evaluation

2.1 Capability Hierarchy and Task Design

We first outline how to construct a domain-specific hierarchy of capabilities. Following the terminology of [10], we refer to the model being evaluated as the *subject* model. To perform a fine-grained evaluation of the subject model's capabilities in a domain, we define a hierarchical structure over the capabilities. In this hierarchy, a domain consists of multiple *areas*, and each area is further divided into *capabilities*. For example, in Mathematics, areas include Algebra, Calculus, Geometry, etc. Within the area of Algebra, capabilities may include Linear Equations, Factoring Expressions, etc. This hierarchy is flexible and can include additional levels. Figure 1a illustrates this structure.

To evaluate a subject model on a given capability, we use a set of *tasks*. Each task consists of a *problem* and a corresponding *reference solution* that serves as the ground truth for scoring. To ensure a robust estimate of a model's performance on a given capability, we evaluate it on a large set of tasks. Capability-level scores are computed by aggregating individual task scores, typically using the mean.

To construct the capability hierarchy and tasks, our framework uses a powerful foundation model termed the *scientist* model (following [10]). The scientist proposes domain areas, decomposes them into capabilities, and generates tasks with reference solutions. To ensure correctness, we introduce a verification step where another model reviews each reference solution. To ensure the reliability of both the generated solutions and the verification step, we conduct a human inspection of the outputs from the task generation and verification processes, which we detail in Section B.3. An abstract overview of the pipeline is shown in Figure 1b.

2.2 Latent Modeling of Capabilities

We assume that capabilities in a domain are specified in a discrete space $\mathcal{T}$. For example, $\mathcal{T}$ could be the text space, where each capability is described by a short natural language statement, such as "linear equations" or "integration by parts." Since function approximation is challenging to perform directly in $\mathcal{T}$, we instead map each capability description to a continuous latent space $\mathcal{Z} \subset \mathbb{R}^d$ using a pretrained text encoder $E : \mathcal{T} \to \mathcal{Z}$. The subject model's performance on a capability $z \in \mathcal{Z}$ is modeled by a surrogate function $f : \mathcal{Z} \to \mathbb{R}^+$, where $f(z)$ denotes the capability score, i.e., how well the model performs on capability z. We refer to f as the *capability function* and assume it to be smooth. This assumption aligns well with real-world LLM behavior, where related capabilities often exhibit correlated performance [11, 12, 13].

A key requirement of our approach is that the encoder E preserves semantic relationships between capabilities, i.e., similar capabilities in $\mathcal{T}$ should be mapped to nearby points in $\mathcal{Z}$. This is crucial for generalization and uncertainty modeling of the capability function f. In Section 3.2, we empirically demonstrate that modern pretrained text encoders satisfy this requirement.

The capability space is large, making exhaustive evaluation infeasible. Even estimating scores on a subset is costly, since each requires generating hundreds of tasks and evaluating the subject model, resulting in substantial API calls. Fixed evaluation sets may also miss important capabilities. To address this, we employ *active learning* to adaptively select and score informative capabilities. In each round, we choose a candidate capability, obtain its score, and update f. We adopt Bayesian optimization

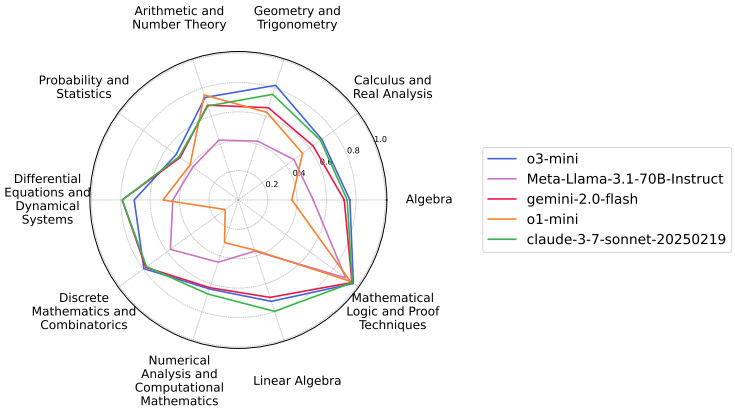


Figure 2: Model scores across different areas in Mathematics. The reported score for each area is the average score of all capabilities within that area.

with Gaussian process (GP) regression, a principled framework widely used for global optimization under limited or expensive data [14, 15, 16, 17, 18]. A GP is denoted by $f(\mathbf{x}) \sim \mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}'))$ [19], where $m(\mathbf{x}) = \mathbb{E}[f(\mathbf{x})]$ and kernel $k(\mathbf{x}, \mathbf{x}') = \mathbb{E}[(f(\mathbf{x}) - m(\mathbf{x}))(f(\mathbf{x}') - m(\mathbf{x}'))]$. GPs support active learning via posterior mean and variance estimates. Two classical approaches leverage posterior variance: (1) selecting the candidate with largest variance to maximize information gain [20], $\mathbf{x}^* = \arg\max_{\mathbf{x} \in \mathcal{U}} \mathbb{V}[f(\mathbf{x})]$, and (2) minimizing expected posterior variance over the input space [21], $\mathbf{x}^* = \arg\min_{\mathbf{x} \in \mathcal{U}} \mathbb{E}_{y|\mathbf{x}} \left[\int \mathbb{V}[f(\mathbf{x}')|\mathcal{D} \cup (\mathbf{x}, y)] d\mathbf{x}' \right]$. Further GP background and details on these methods are provided in Appendix A.

The embedding space of pretrained encoder is usually high-dimensional (e.g., 512-dimensional), making it challenging to perform regression. Hence, we apply a dimensionality reduction technique like t-SNE [22] or Principal Component Analysis (PCA). The resulting low-dimensional representations enable efficient active learning: In each round we compute acquisition scores for all candidate capabilities, select the optimal candidate, and assess the subject model's performance on it. The newly obtained (capability, score) pair then updates the GP model. Algorithm 1 in the Appendix formalizes this procedure.

3 Experiments

We evaluate ACE in the domain of Mathematics. All experiments use OpenAI's `o4-mini`[1] as the scientist model. We first prompt it to generate broad areas, then specific capabilities in a modified `METR`[2] format following [10], each with a `name`, `description`, and `Python class` specifying exemplar tasks, instructions, and scoring. Prompts are given in Appendix H.1 and H.2. For each capability we run the task generation pipeline to produce diverse problems. Each problem is solved using the capability's instructions, and solutions are verified by an additional LLM pass to filter errors. The same `o4-mini` model is used for all task generation and verification (prompts in Appendix H.3). For evaluation, we adopt the `Inspect` framework [9], which executes tasks using capability-specific instructions and scoring. We use binary scoring: a solution is correct if it matches the ground truth (score=1) , otherwise incorrect (score=0). This procedure yields a benchmark of 78 capabilities across 10 areas, with 8,529 verified tasks. Full capability lists and scores are in Appendix D.2.

3.1 Benchmarking LLMs in Mathematics

To demonstrate ACE's utility independent of active learning, we evaluate open- and closed-source models on all 78 mathematical capabilities. This establishes ACE's ability to generate high-quality, domain-specific benchmarks supporting both broad and fine-grained assessment. Capability scores are averaged over tasks, and area scores over capabilities. Figure 2 reports performance of five

[1] https://platform.openai.com/docs/models/o4-mini
[2] https://metr.org/

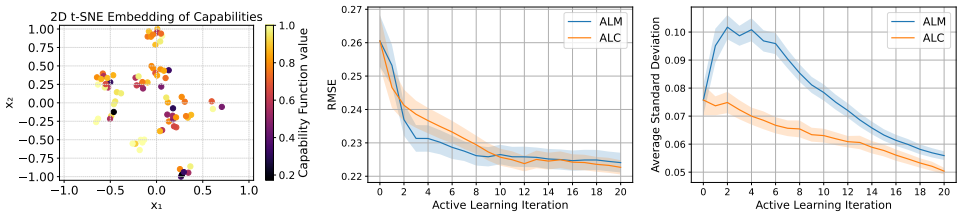


Figure 3: Performance of active learning acquisition techniques. The left column shows the ground truth values. The center column shows test error (root mean squared error), and the right column shows test set average posterior standard deviation. Shaded areas indicate 95% confidence interval.

models across areas. Fine-grained capability-level results appear in Appendix D.2, with full results in Appendix C. These results highlight the value of structured, capability-based evaluation: even strong models exhibit distinct area-level strengths and weaknesses not visible in aggregate metrics.

3.2 Semantic Relationships of Capabilities in Latent Space

Reliable approximation of the capability function $f(z)$ depends on whether the latent space $\mathcal{Z}$ preserves semantic relationships between capabilities. In particular, capabilities within the same area should be embedded close to each other in $\mathcal{Z}$. This structure facilitates generalization and smooth function approximation. Two components influence the structure of the latent space: the text encoder, which maps natural language descriptions of capabilities to high-dimensional embeddings, and the dimensionality reduction technique used to project these embeddings into a lower-dimensional space. We study the effect of the text encoder in isolation and combined effect of the text encoder and dimensionality reduction on preserving the semantic relationship between capabilities. Detailed results are available in Appendix D.1. Our findings demonstrate that a strong text encoder paired with an appropriate dimensionality reduction method can effectively preserve semantic relationships between capabilities, which enables active learning and approximation of the capability function in a low-dimensional latent space.

3.3 Evaluating Active Learning Acquisition Functions

In Section 2.2, we introduced two variance-based acquisition functions for active learning with Gaussian processes: the MacKay's method [20] (`ALM`), and the Cohn's method [21] (`ALC`). We apply these strategies to the capability function approximation problem in Mathematics. The data comes from scoring the `o3-mini` subject model across 78 capabilities as described in Section 3.1. The full dataset is $\{(z_i, s_i)\}_{i=1}^{78}$, where z_i is the 2D t-SNE embedding of the i-th capability and s_i is the corresponding capability score. The dataset is split equally into training and test sets. The GP model is initialized with two randomly selected capabilities, and active learning is performed on the remaining training set. Figure 3 shows test set metrics. While both `ALM` and `ALC` show a similar trend in reducing RMSE, `ALC` is capable of reducing uncertainty more rapidly. We conducted this experiment with other subject models and observed similar results, which are available in Appendix F. These results demonstrate that active learning in the capability latent space, particularly the `ALC` variant, can be effective for learning the capability function.

4 Conclusion

This paper introduces ACE, a framework for scalable and structured evaluation of foundation models. ACE leverages the generative power of LLMs to construct semantically meaningful capability hierarchies and associated evaluation tasks for a target domain. It further employs active learning in a latent semantic space to efficiently estimate a model's capability function and uncover strengths and weaknesses with minimal evaluation cost. A limitation of ACE is its reliance on a single scientist model. This raises questions about systematic biases in the evaluation process. To mitigate this we can employ a panel of diverse models to jointly generate and verify tasks, thereby reducing model-specific biases. As foundation models are increasingly deployed, the need for fine-grained, dynamic, and cost-effective evaluation will also grow. By integrating strong generative models with active learning, our framework lays the foundation for a robust and reliable evaluation paradigm.

References

[1] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

[2] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems, 2021. *URL https://arxiv. org/abs/2110.14168*, 9, 2021.

[3] Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. Drop: A reading comprehension benchmark requiring discrete reasoning over paragraphs. *arXiv preprint arXiv:1903.00161*, 2019.

[4] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*, 2020.

[5] Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. *NeurIPS*, 2021.

[6] Long Phan, Alice Gatti, Ziwen Han, Nathaniel Li, Josephina Hu, Hugh Zhang, Chen Bo Calvin Zhang, Mohamed Shaaban, John Ling, Sean Shi, et al. Humanity's last exam. *arXiv preprint arXiv:2501.14249*, 2025.

[7] Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, Abu Awal Md Shoeb, Abubakar Abid, Adam Fisch, Adam R Brown, Adam Santoro, Aditya Gupta, Adrià Garriga-Alonso, et al. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *arXiv preprint arXiv:2206.04615*, 2022.

[8] Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? *arXiv preprint arXiv:1905.07830*, 2019.

[9] AI Security Institute, UK. Inspect AI: Framework for Large Language Model Evaluations. https://github.com/UKGovernmentBEIS/inspect_ai, 2024. Accessed: 2024-05.

[10] Cong Lu, Shengran Hu, and Jeff Clune. Automated capability discovery via model self-exploration. *arXiv preprint arXiv:2502.07577*, 2025.

[11] Xiaoqiang Wang, Lingfei Wu, Tengfei Ma, and Bang Liu. Fac^{2}e: Better understanding large language model capabilities by dissociating language and cognition. *arXiv preprint arXiv:2403.00126*, 2024.

[12] Charlotte Siska, Katerina Marazopoulou, Melissa Ailem, and James Bono. Examining the robustness of llm evaluation to the distributional assumptions of benchmarks. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10406–10421, 2024.

[13] David Ilić and Gilles E Gignac. Evidence of interrelated cognitive-like capabilities in large language models: Indications of artificial general intelligence or achievement? *Intelligence*, 106:101858, 2024.

[14] Gustavo Malkomes. Automating active learning for gaussian processes. 2019.

[15] Dirk Gorissen, Karel Crombecq, Ivo Couckuyt, and Tom Dhaene. Automatic approximation of expensive functions with active learning. *Foundations of Computational, Intelligence Volume 1: Learning and Approximation*, pages 35–62, 2009.

[16] Siwei Fu. Active learning for solving expensive optimization problems, 2022.

[17] Christoffer Riis, Francisco Antunes, Gérald Gurtner, Francisco Camara Pereira, Luis Delgado, and Carlos M Lima Azevedo. Active learning metamodels for atm simulation modeling. In *11th SESAR Innovation Days*, 2021.

[18] Sylvain Chabanet, Hind Bril El-Haouzi, and Philippe Thomas. Coupling digital simulation and machine learning metamodel through an active learning approach in industry 4.0 context. *Computers in Industry*, 133:103529, 2021.

[19] Carl Edward Rasmussen and Christopher KI Williams. *Gaussian processes for machine learning*, volume 2. MIT press Cambridge, MA, 2006.

[20] David JC MacKay. Information-based objective functions for active data selection. *Neural computation*, 4(4):590–604, 1992.

[21] David A Cohn. Neural network exploration using optimal experiment design. *Neural networks*, 9(6):1071–1083, 1996.

[22] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.

[23] Sambu Seo, Marko Wallat, Thore Graepel, and Klaus Obermayer. Gaussian process regression: Active data selection and test point rejection. In *Mustererkennung 2000*, pages 27–34. Springer, 2000.

[24] Matthias Seeger. Pac-bayesian generalisation error bounds for gaussian process classification. *Journal of machine learning research*, 3(Oct):233–269, 2002.

[25] Emilie Kaufmann, Olivier Cappé, and Aurélien Garivier. On bayesian upper confidence bounds for bandit problems. In *Artificial intelligence and statistics*, pages 592–600. PMLR, 2012.

[26] Percy Liang, Rishi Bommasani, Tony Lee, Dimitris Tsipras, Dilara Soylu, Michihiro Yasunaga, Yian Zhang, Deepak Narayanan, Yuhuai Wu, Ananya Kumar, et al. Holistic evaluation of language models. *arXiv preprint arXiv:2211.09110*, 2022.

[27] Stephanie Lin, Jacob Hilton, and Owain Evans. Truthfulqa: Measuring how models mimic human falsehoods, 2022.

[28] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding, 2021.

[29] Chunyuan Deng, Yilun Zhao, Xiangru Tang, Mark Gerstein, and Arman Cohan. Investigating data contamination in modern benchmarks for large language models, 2024.

[30] Marco Tulio Ribeiro and Scott Lundberg. Adaptive testing and debugging of NLP models. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio, editors, *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3253–3267, Dublin, Ireland, May 2022. Association for Computational Linguistics.

[31] Douwe Kiela, Max Bartolo, Yixin Nie, Divyansh Kaushik, Atticus Geiger, Zhengxuan Wu, Bertie Vidgen, Grusha Prasad, Amanpreet Singh, Pratik Ringshia, et al. Dynabench: Rethinking benchmarking in nlp. *arXiv preprint arXiv:2104.14337*, 2021.

[32] Zhehao Zhang, Jiaao Chen, and Diyi Yang. Darg: Dynamic evaluation of large language models via adaptive reasoning graph, 2024.

[33] Zhiyuan Zeng, Yizhong Wang, Hannaneh Hajishirzi, and Pang Wei Koh. Evaltree: Profiling language model weaknesses via hierarchical capability trees, 2025.

[34] Yongliang Shen, Kaitao Song, Xu Tan, Wenqi Zhang, Kan Ren, Siyu Yuan, Weiming Lu, Dongsheng Li, and Yueting Zhuang. Taskbench: Benchmarking large language models for task automation, 2024.

[35] Xiang Lisa Li, Farzaan Kaiyom, Evan Zheran Liu, Yifan Mai, Percy Liang, and Tatsunori Hashimoto. Autobencher: Towards declarative benchmark construction, 2025.

[36] Yang Li, Jie Ma, Miguel Ballesteros, Yassine Benajiba, and Graham Horwood. Active evaluation acquisition for efficient llm benchmarking, 2024.

[37] Shengran Hu, Cong Lu, and Jeff Clune. Automated design of agentic systems, 2025.

[38] Han Jiang, Xiaoyuan Yi, Zhihua Wei, Ziang Xiao, Shu Wang, and Xing Xie. Raising the bar: Investigating the values of large language models via generative evolving testing, 2025.

[39] Davis Brown, Prithvi Balehannina, Helen Jin, Shreya Havaldar, Hamed Hassani, and Eric Wong. Adaptively evaluating models with task elicitation, 2025.

[40] Mantas Mazeika, Long Phan, Xuwang Yin, Andy Zou, Zifan Wang, Norman Mu, Elham Sakhaee, Nathaniel Li, Steven Basart, Bo Li, David Forsyth, and Dan Hendrycks. Harmbench: A standardized evaluation framework for automated red teaming and robust refusal, 2024.

[41] Sabit Hassan, Anthony Sicilia, and Malihe Alikhani. Active learning for robust and representative llm generation in safety-critical scenarios, 2024.

[42] Michelle Yuan, Hsuan-Tien Lin, and Jordan Boyd-Graber. Cold-start active learning through self-supervised language modeling. In Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu, editors, *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7935–7948, Online, November 2020. Association for Computational Linguistics.

[43] Katerina Margatina, Giorgos Vernikos, Loïc Barrault, and Nikolaos Aletras. Active learning by acquiring contrastive examples. In Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih, editors, *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 650–663, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics.

Appendix

A Active Learning with Gaussian Processes

A Gaussian process (GP) is a collection of random variables, any finite number of which have a joint Gaussian distribution [19]. It is fully specified by a mean function $m(\mathbf{x}) = \mathbb{E}[f(\mathbf{x})]$ and a covariance (kernel) function $k(\mathbf{x}, \mathbf{x}') = \mathbb{E}[(f(\mathbf{x}) - m(\mathbf{x}))(f(\mathbf{x}') - m(\mathbf{x}'))]$:

$$f(\mathbf{x}) \sim \mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}'))$$

Consider a regression task with training data $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$ where $y_i = f(\mathbf{x}_i) + \epsilon_i$ with $\epsilon_i \sim \mathcal{N}(0, \sigma_n^2)$. For a test input $\mathbf{x}_*$, the predictive distribution is Gaussian:

$$p(f_*|\mathbf{x}_*, \mathcal{D}) = \mathcal{N}(\mathbb{E}[f_*], \mathbb{V}[f_*]),$$

with predictive mean and variance:

$$\mathbb{E}[f_*] = \mathbf{k}_*{}^\top(\mathbf{K} + \sigma_n^2\mathbf{I})^{-1}\mathbf{y} \quad (1)$$

$$\mathbb{V}[f_*] = k(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{k}_*{}^\top(\mathbf{K} + \sigma_n^2\mathbf{I})^{-1}\mathbf{k}_*, \quad (2)$$

in which $\mathbf{K}$ is the kernel matrix with $K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$, $\mathbf{y} = \{y_1, \ldots, y_N\}$, and $\mathbf{k}_* = [k(\mathbf{x}_1, \mathbf{x}_*), ..., k(\mathbf{x}_N, \mathbf{x}_*)]^\top$.

The function-space view interprets the GP as defining a distribution over functions, where the kernel function encodes prior assumptions such as smoothness. A common choice is the squared exponential kernel:

$$k(\mathbf{x}, \mathbf{x}') = \sigma_f^2 \exp\left(-\frac{||\mathbf{x} - \mathbf{x}'||^2}{2l^2}\right).$$

GPs naturally lend themselves to active learning due to the availability of posterior mean and variance estimates. In particular two well-known approaches leverage GP posterior variance for active learning. [20] aims at maximizing the expected information gain by selecting the data where the model has maximum variance. This is performed by selecting points that maximize the posterior variance:

$$\mathbf{x}^* = \arg\max_{\mathbf{x} \in \mathcal{U}} \mathbb{V}[f(\mathbf{x})], \quad (3)$$

where $\mathcal{U}$ is the pool of unlabeled candidates. This is equivalent to maximizing the reduction in entropy H of the GP posterior:

$$\mathbf{x}^* = \arg\max_{\mathbf{x} \in \mathcal{U}} H[p(f|\mathcal{D})] - \mathbb{E}_{y|\mathbf{x}}[H[p(f|\mathcal{D} \cup (\mathbf{x}, y))]].$$

It is possible to perform optimization of Eq. 2 with respect to $\mathbf{x}^*$ using, e.g., gradient ascent [23].

The second method is motivated by minimizing the generalization error in terms of mean squared error (MSE). Using the bias-variance decomposition of MSE and making some assumptions with respect to the magnitude of bias, it can be shown that minimizing MSE can be approximated by choosing the candidate point that reduces the expected predictive variance over the entire input space [21]:

$$\mathbf{x}^* = \arg\min_{\mathbf{x} \in \mathcal{U}} \mathbb{E}_{y|\mathbf{x}}\left[\int \mathbb{V}[f(\mathbf{x}')|\mathcal{D} \cup (\mathbf{x}, y)]d\mathbf{x}'\right] \quad (4)$$

In practice the integration in Eq. 4 can be approximated by Monte Carlo or by calculating the variance over a holdout set.

For GPs, both approaches can be approximated efficiently as the posterior covariance matrix can be updated incrementally using rank-1 updates [24]. The active learning process iteratively fits the GP to current labeled data, $\mathcal{L}$, computes the acquisition score (Eq. 3 or 4) for all $\mathbf{x} \in \mathcal{U}$, selects $\mathbf{x}^*$ that maximizes the acquisition score, queries for y^* at $\mathbf{x}^*$, and updates the labeled and candidate sets, $\mathcal{L} \leftarrow \mathcal{L} \cup \{(\mathbf{x}^*, y^*)\}, \mathcal{U} \leftarrow \mathcal{U} \setminus \{\mathbf{x}^*\}$.

Algorithm 1: Active Capability Learning

Input:

Initial capability set $\mathcal{C} = \{c_i\}_{i=1}^{N}$ generated by the scientist model
Pretrained encoder $E : \mathcal{C} \to \mathbb{R}^d$
Dimensionality reduction method φ (e.g., PCA, t-SNE)
Evaluation module `Evaluate()` to score a capability
Active learning acquisition function $\alpha(\cdot)$
Target latent dimension $d' \ll d$

Initialization:

1. Encode all capabilities: $\mathbf{Z} = \{E(c_i) | c_i \in \mathcal{C}\}$
2. Reduce dimensionality: $\mathbf{Z}' = \varphi(\mathbf{Z}) \in \mathbb{R}^{N \times d'}$
3. Initialize training set $\mathcal{D}$ by randomly selecting a small number of capabilities (e.g., 2) from $\mathcal{C}$ and scoring them using `Evaluate()`

`// Active learning`
while *stopping conditions not met* **do**
  1. Fit GP model, f, on current $\mathcal{D}$ (non-parametric)
  2. Compute acquisition scores: $\forall \mathbf{z}'_i \in \mathbf{Z}' \setminus \mathcal{D}, \alpha_i \leftarrow \alpha(\mathbf{z}'_i; f)$
  3. Select the best candidate: $j \leftarrow \arg\max_i \alpha_i$
  4. Obtain capability score: $s_j \leftarrow \texttt{Evaluate}(c_j)$
  5. Update training set: $\mathcal{D} \leftarrow \mathcal{D} \cup \{(\mathbf{z}'_j, s_j)\}$
end
return $\mathcal{D}$

B Additional Experiments

B.1 Scientist Model Selection

We evaluate leading OpenAI LLMs on 100 random tasks (or samples) from the MATH dataset [5] and select the LLM with the best performance. Given that o3-mini, o3, and o4-mini have comparable scores (as shown in Table 1), we choose o4-mini since it is the most recent and cost-effective option.

	gpt-4o	gpt-4.1	o3-mini	o3	o4-mini
MATH (100 tasks)	0.88	0.89	0.96	0.96	0.95

Table 1: Performance of leading OpenAI LLMs on 100 random tasks from the MATH dataset.

B.2 Performance Distribution Across MATH and Generated Benchmarks

A key requirement for automating capability evaluation is that the tasks generated by the scientist model be both valid and discriminative with respect to subject model performance. To assess this, we conduct an experiment that compares a subject model performance on generated tasks to its performance on tasks from a human-curated benchmark dataset. We use the MATH dataset [5], which contains 12,500 high school competition-level problems labeled by *area*. There are seven major capability categories (areas) in MATH: Pre-algebra, Algebra, Number Theory, Counting and Probability, Geometry, Intermediate Algebra, and Pre-calculus. These categories serve as the capabilities for our comparison.

For each capability (area), we extract the corresponding subset of problems from MATH and evaluate the subject model's performance on them. Next, we prompt the scientist model to generate new tasks targeting the same capability. To ensure diversity and coverage, we instruct the scientist model (`o4-mini`) to generate problems of varying difficulty within each capability. We then evaluate the subject models on these problems. For each capability, we take the average of scores across all tasks (problems) to form the capability score. Results are shown in Table 2. In addition, Figure 4 shows the scatter plot of model performance on the MATH dataset capabilities vs synthetic tasks.

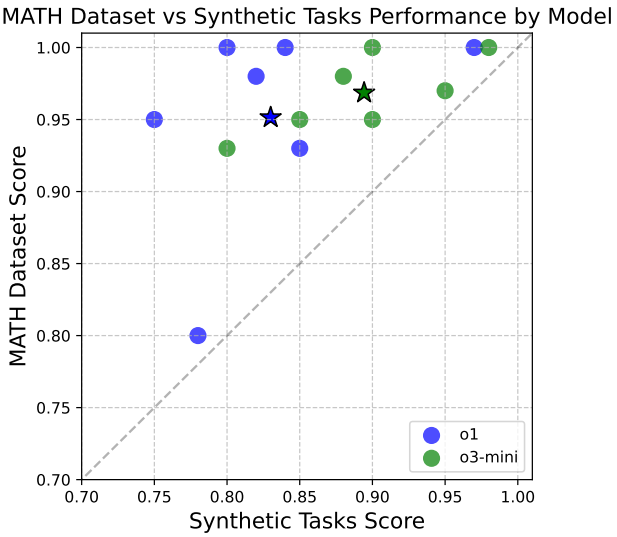


Figure 4: Comparing performance of models on the MATH dataset vs synthetic tasks. Stars indicate average score across all capabilities.

For both subject models, comparing the distribution of capability scores across MATH and synthetic tasks reveals greater variation in capability scores on the synthetic benchmark. This suggests that our generated tasks span a broader range of problem types and difficulties within each capability. Consequently, the synthetic dataset provides a more nuanced and discriminative assessment of model strengths and weaknesses. These results support the viability of our framework as an effective tool for evaluating foundation models in a given domain, offering a potentially more revealing alternative to static, human-curated benchmarks.

Capability (Area)	o1		o3-mini	
	Synthetic	MATH	Synthetic	MATH
Algebra	0.84	1.00	0.88	0.98
Counting & prob.	0.82	0.98	0.98	1.00
Geometry	0.85	0.93	0.90	0.95
Intermediate Algebra	0.75	0.95	0.85	0.95
Number Theory	0.80	1.00	0.90	1.00
Pre-algebra	0.97	1.00	0.95	0.97
Pre-calculus	0.78	0.80	0.80	0.93
Average score	0.83	0.95	0.89	0.97

Table 2: Comparison of Synthetic vs MATH task scores for capabilities of the MATH dataset using o4-mini as the scientist model.

B.3 Manual Inspection of Tasks

To evaluate the quality of the task generation pipeline and the reliability of the automated verification step, we conducted a manual inspection of a subset of tasks. Specifically, we randomly selected 12 capabilities across three mathematical areas.[3] For each capability, we sampled 15 tasks, resulting in a total of 180 problem–solution pairs. Each task's problem, solution, and verification model output were manually reviewed by solving the problem and comparing the correct solution to the automated verification outcome.

[3]Areas include: (1) Geometric and Spatial Reasoning, (2) Statistical and probabilistic analysis, and (3) Symbolic and Analytical Mathematics. Capabilities include: Bayesian inference, Cross-section area, Eigen problems, Hypothesis testing, Markov chain analysis, Multivariate integration, Multivariate distribution analysis, ODE solving, Polyhedron net identification, Spatial puzzle cubes, Taylor series, and Transformational geometry.

The results indicate a high degree of agreement between human and automated verification. Of the 180 tasks, we observed the following confusion matrix: True Positives = 158, False Negatives = 14, False Positives = 1, and True Negatives = 7. This corresponds to a precision of **99.4%**, recall of **91.9%**, and overall verification accuracy of **91.7%**. These results support the conclusion that the automated pipeline for task generation and verification is reliable for evaluating model capabilities at scale. Despite strong performance in task generation, our inspection surfaced a few recurring issues that are important to address in future iterations of the framework:

1. **Rounding Errors.** Infrequent but notable rounding inaccuracies occurred when intermediate numerical results were used in subsequent calculations. These rounding issues sometimes led to small deviations in final answers and highlight the need for improved numerical precision handling.
2. **Lack of Task Diversity.** Many tasks within a capability were structurally or conceptually similar. Increasing task diversity—across difficulty levels and subtopics—would yield a more comprehensive assessment of model performance.
3. **Inter-Task Dependencies.** Since multiple tasks were generated from a single prompt (to minimize repetition), some questions inadvertently referenced earlier tasks. Future prompts should explicitly enforce task independence to avoid this issue.
4. **Parsing Limitations.** Some task-solving instructions required the model to output the final answer after an "ANSWER" keyword. The current parsing logic does not support multi-line answers, which can result in incomplete ground truth extraction and premature task rejection during verification. Improving parsing robustness would reduce unnecessary filtering of valid tasks.

Addressing these issues will further enhance the robustness and reliability of automated task generation and verification.

B.4 Evaluating Active Learning Acquisition Functions

In Section 2.2, we introduced two variance-based acquisition functions for active learning with Gaussian processes. MacKay's method [20], denoted by `ALM`, selects the candidate with the highest posterior variance. Cohn's method [21], denoted by `ALC`, selects the candidate expected to yield the largest overall reduction in posterior variance across the input space. In our implementation, this is approximated by averaging posterior variance over the entire set of unused candidate points.

We first evaluate these acquisition strategies on a synthetic toy problem: approximating the target function $f(x) = \sin(20\pi x)\exp(-5x)$ over the interval $x \in [0, 1]$. The candidate set consists of 32 points sampled uniformly from the domain with a noisy label $y = f(x) + \epsilon$, where $\epsilon \sim \mathcal{N}(0, 0.1)$. The GP model is initially fit on two randomly selected candidates; the remaining points are used for active learning. We repeat the experiment 200 times to compute confidence intervals for both root mean squared error (RMSE) and average posterior standard deviation. These metrics are evaluated on a test set of 1000 equidistant points. Results, shown on the top row of Figure 5, demonstrate that `ALC` outperforms `ALM`, achieving more rapid reductions in both prediction error and model uncertainty.

Next, we apply the same strategies to the capability function approximation problem in Mathematics. The data comes from scoring the `o3-mini` model across 78 capabilities as described in Section 3.1. The full dataset is $\{(z_i, s_i)\}_{i=1}^{78}$, where z_i is the 2D t-SNE embedding of the i-th capability and s_i is the corresponding capability score of `o3-mini`. The dataset is split equally into training and test sets. As before, the GP model is initialized with two randomly selected capabilities, and active learning is performed on the remaining training set. Figure 5 (bottom row) shows test set metrics. While both `ALM` and `ALC` show a similar trend in reducing RMSE, `ALC` is capable of reducing uncertainty more rapidly. We conducted this experiment with other subject models and observed similar results, which are available in Appendix F.

While this experiment used a relatively small set of capabilities, the results demonstrate that active learning in the capability latent space, particularly using total variance reduction as the acquisition function (`ALC`), can be effective when the latent space preserves semantic structure. Active learning is especially valuable when the number of capabilities is large and exhaustive evaluation is infeasible due to budget or computational constraints. Although our focus here was on variance-based acquisition functions for approximating the capability function, other acquisition strategies can be chosen to

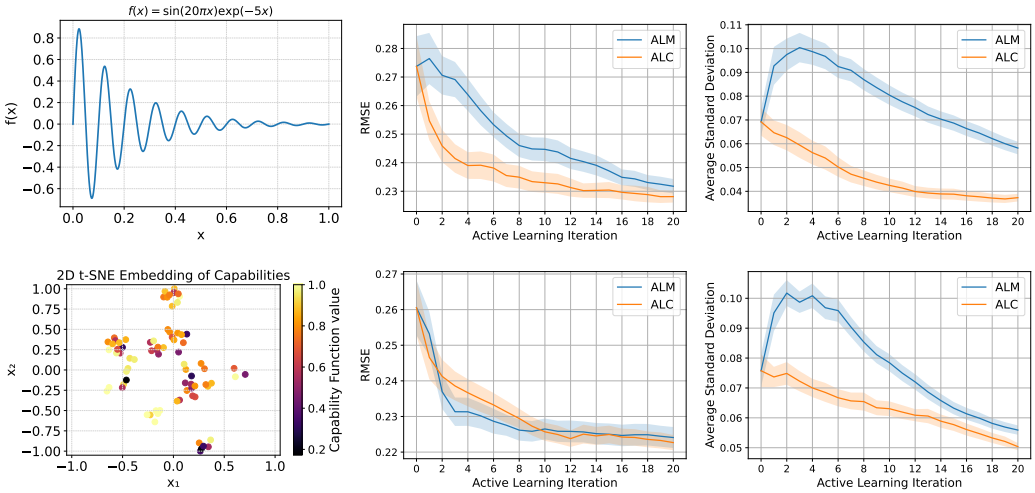


Figure 5: Performance of active learning acquisition techniques. **Top:** the toy problem, **bottom:** the Mathematics capability dataset with `o3-mini` as the subject model. The left column shows the ground truth function and values. The center column plots show test error (root mean squared error), and the right column plots show test set average posterior standard deviation. Shaded areas indicate 95% confidence interval.

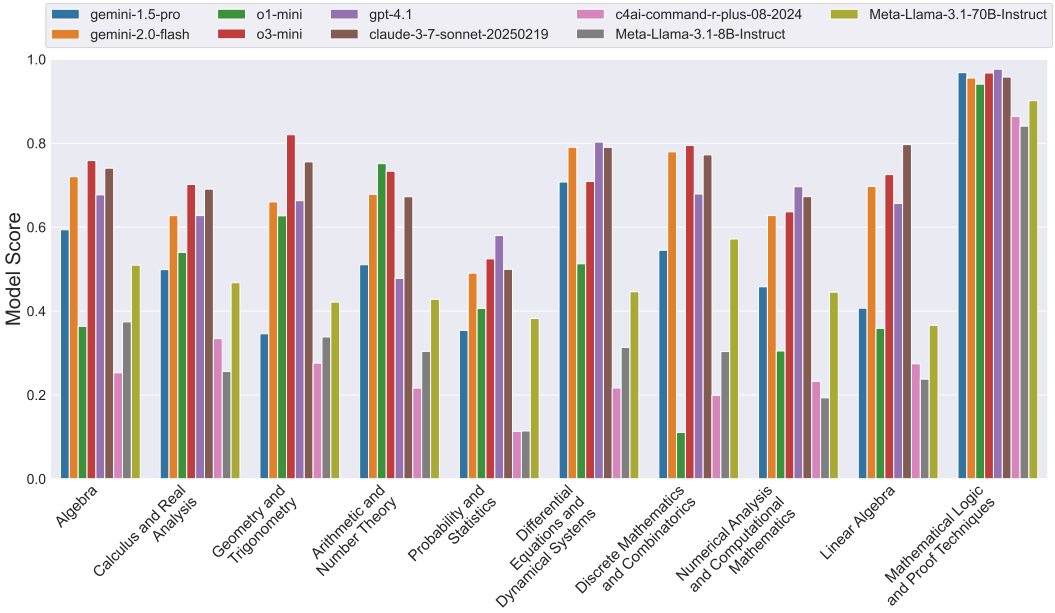


Figure 6: Average scores of 9 LLMs on 10 areas.

meet other objectives. For example, if the goal is to identify the capabilities on which a subject model excels, the Upper Confidence Bound (UCB) acquisition function [25] may be a more appropriate choice.

C Additional Benchmark Results

Figure 6 shows the average LLM scores across 10 areas for 9 LLMs.

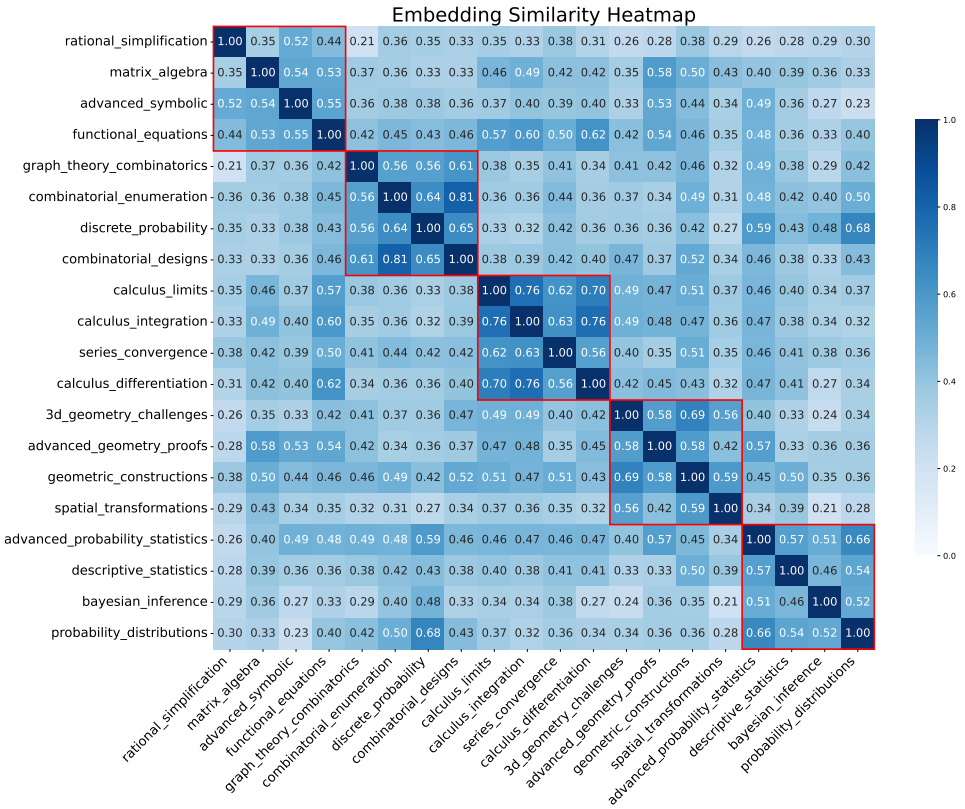


Figure 7: The heatmap illustrating the cosine similarity matrix of capability embeddings. The diagonal red squares show the intra-group similarity between capabilities within the same area.

D Capability Details

In this section, we provide details on the generated capabilities, their embeddings used in our method, and LLM scores evaluated on each capability.

D.1 Studying the Semantic Relationship of Capabilities in Latent Space

We first study the effect of the text encoder in isolation. Using the OpenAI `text-embedding-3-small` model[4] (512-dimensional output), we embed a subset of 20 capabilities sampled from 5 mathematical areas. Each embedding is generated by concatenating the `name`, `area`, and `description` fields of the capability as input to the encoder. We compute the pairwise cosine similarity matrix between capabilities. As the heatmap in Figure 7 shows, capability embeddings within the same area have higher cosine similarity compared to the capabilities in other areas. The results demonstrate that the encoder meaningfully captures semantic similarity between capabilities.

Next, we assess the combined effect of the text encoder and dimensionality reduction. We embed all 78 capabilities using the same encoder and project the resulting representations into a 2D latent space using either t-SNE or PCA. Figure 8 shows the resulting distributions. Both techniques preserve coarse structure, but t-SNE produces more distinct clusters for capabilities within each area.

D.2 Capability Scores

In this section, Table 3 details the full list of capabilities with their respective areas.

[4] https://platform.openai.com/docs/guides/embeddings/

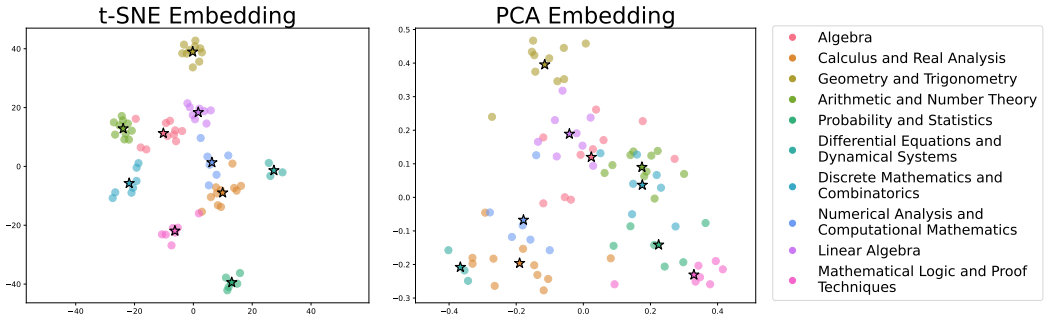


Figure 8: Two-dimensional representation of Mathematics capabilities using t-SNE (left) and PCA (right). Each point corresponds to a capability, and colors indicate high-level areas. Stars indicate the mean of capability representations for each area.

Area	Capability Name	claude-3-7 -sonnet	o3-mini	gemini-2.0 -flash	o1-mini	Meta-Llama-3.1 -70B-Instruct
Algebra	abstract algebra	0.91	0.97	0.92	0.09	0.87
	algebraic anequalities	0.81	0.76	0.85	0.64	0.50
	complex algebra	0.75	0.57	0.66	0.23	0.61
	exponential and logarithmic equations	0.45	0.50	0.37	0.19	0.28
	functional equations	0.59	0.55	0.46	0.38	0.20
	nested radical simplification	0.56	0.86	0.61	0.68	0.32
	nonlinear systems	0.85	0.88	0.95	0.57	0.60
	parameter conditions	0.64	0.61	0.57	0.40	0.37
	symmetric polynomials	0.90	0.96	0.83	0.01	0.48
	polynomial factorization	0.95	0.94	0.97	0.44	0.87
Calculus and Real Analysis	advanced improper integrals	0.52	0.54	0.47	0.41	0.36
	calculus of variations	0.89	0.87	0.70	0.89	0.44
	epsilon delta limits	0.41	0.58	0.36	0.58	0.18
	fourier series analysis	0.44	0.75	0.78	0.62	0.41
	integral calculus	0.87	0.81	0.85	0.58	0.70
	lebesgue integration	0.39	0.46	0.13	0.69	0.05
	limits and continuity	0.67	0.64	0.47	0.23	0.49
	multivariable calculus	0.93	0.77	0.83	0.44	0.71
	real analysis proofs	0.81	0.78	0.80	0.65	0.44
	series and sequences	0.76	0.66	0.66	0.27	0.55
	single variable differentiation	0.90	0.88	0.86	0.57	0.81
Geometry and Trigonometry	analytic geometry circles	0.84	0.80	0.80	0.59	0.66
	circle inversion geometry	0.52	0.75	0.46	0.76	0.03
	complex plane geometry	0.44	0.77	0.47	0.71	0.13
	conic sections properties	0.85	0.88	0.83	0.88	0.50
	geometric transformations	0.87	0.87	0.88	0.16	0.66
	projective geometry	0.59	0.58	0.33	0.81	0.02
	three d geometry volumes	0.98	1.00	0.97	0.34	0.59
	trigonometric equations nonlinear	0.80	0.87	0.32	0.87	0.37
	trigonometric identities simplification	0.94	0.96	0.95	0.58	0.83
	vector geometry	0.73	0.73	0.60	0.56	0.42
Arithmetic and Number Theory	arithmetic functions	0.94	0.87	0.78	0.97	0.68
	chinese remainder	0.65	0.86	0.39	0.95	0.03
	continued fractions	0.28	0.56	0.56	0.59	0.24
	egyptian fractions	0.53	0.73	0.61	0.70	0.05
	gcd lcm valuations	0.78	0.80	0.91	0.93	0.73

Continued on next page

Area	Capability Name	claude-3-7 -sonnet	o3-mini	gemini-2.0 -flash	o1-mini	Meta-Llama-3.1 -70B-Instruct
	modular exponentiation	0.30	0.65	0.56	0.72	0.03
	prime_factorization	0.86	0.98	0.86	0.94	0.56
	quadratic residues	0.67	0.58	0.61	0.66	0.53
	special numbers	0.96	1.00	0.95	1.00	0.73
	zeckendorf representation	0.75	0.31	0.56	0.06	0.70
Probability and Statistics	bayesian inference	0.30	0.32	0.28	0.13	0.20
	distribution moments	0.98	0.97	0.91	0.93	0.87
	markov chain probabilities	0.52	0.54	0.55	0.43	0.30
	probability paradoxes	0.71	0.78	0.74	0.41	0.50
	statistical hypothesis testing	0.16	0.19	0.10	0.19	0.12
	survival analysis	0.33	0.35	0.37	0.35	0.30
Differential Equations and Dynamical Systems	boundary value eigenvalue problems	0.81	0.69	0.83	0.40	0.29
	nonlinear systems lyapunov	0.59	0.45	0.55	0.20	0.09
	second order homogeneous linear ode	0.98	0.99	0.98	0.93	0.95
Discrete Mathematics and Combinatorics	combinatorial designs	0.86	0.56	0.88	0.37	0.72
	eulerian trail	0.98	1.00	0.94	0.05	0.75
	generating functions	0.86	0.95	0.88	0.02	0.58
	graph coloring	0.88	1.00	1.00	0.02	0.62
	inclusion exclusion	0.88	0.92	0.83	0.03	0.53
	recurrence relations	0.93	0.97	0.89	0.18	0.79
	stirling numbers second kind	0.02	0.17	0.04	0.11	0.01
Linear Algebra	eigenvalues and eigenvectors	0.89	0.68	0.64	0.33	0.43
	jordan normal form	0.90	0.84	0.87	0.55	0.33
	matrix inversion and determinant	0.88	0.84	0.96	0.19	0.82
	orthogonality and projections	0.82	0.79	0.82	0.39	0.58
	quadratic forms	0.43	0.33	0.26	0.13	0.04
	rank and nullspace	0.99	0.80	0.69	0.76	0.03
	singular value decomposition	0.66	0.73	0.68	0.10	0.20
	vector spaces and subspaces	0.81	0.80	0.66	0.42	0.50
Math Logic and Proof Techniques	epsilon delta proofs	0.85	0.86	0.86	0.67	0.48
	model construction	1.0	1.0	1.0	1.0	1.0
	natural deduction proofs	1.0	1.0	1.0	1.0	1.0
	predicate logic formalization	1.0	1.0	1.0	1.0	1.0
	proof by contradiction	1.0	1.0	1.0	1.0	1.0
	proof by induction	1.0	1.0	1.0	1.0	1.0
	propositional logic translation	0.85	0.91	0.83	0.92	0.83
Numerical Analysis and Computational Mathematics	conditioning and stability	0.64	0.45	0.60	0.39	0.41
	eigenvalue methods	0.86	0.95	0.83	0.31	0.83
	fast fourier transform	0.71	0.79	0.88	0.25	0.6
	monte carlo error analysis	0.60	0.5	0.44	0.30	0.37
	numerical integration	0.46	0.3	0.16	0.09	0.18
	numerical optimization	0.77	0.83	0.85	0.48	0.29

E Compute Resources

We used paid APIs to access the scientist LLM as well as subject LLMs from OpenAI, Google, and Anthropic. Open source models that are used as subject LLMs are hosted on internal cluster. Specifically, we used 1 A40 GPU for Meta-Llama-3.1-8B-Instruct, 2 A100s GPUs for Meta-Llama-3.1-70B-Instruct, and 4 A100s for Cohere Command R+, with 10 GPU hours per GPU for all these open source models.

F Complete Results for Evaluating Active Learning Acquisition Functions

Section B.4 compared the performance of ALC and ALM acquisition functions using `o3-mini` as the subject model. Figure 9 presents additional results for other subject models.

G Related Work

There is a clear trend toward automating and scaling up LLM evaluation. Traditional benchmarks like BIG-bench [7], HELM [26], TruthfulQA [27], and MMLU [28] established the importance of diverse, rigorous evaluation, but are inherently static, limited to predefined tasks, and suffer from data contamination [29]. Automated evaluation approaches range from model-assisted test generation [30, 31] to automatic evaluation with graph and tree construction [32, 33, 34, 30] to fully autonomous task discovery and generation [10, 35], uncovering failures and interesting capabilities that a fixed benchmark might miss.

Most *automatic evaluation* approaches rely on heuristics or predefined objectives like "difficulty", "safety issues", or "knowledge gaps" [35, 33, 36] or objectives defined by evaluation functions [37], maximizing these objectives by probing the areas of prior mistakes. Some methods focus specifically on automatic assessment of ethical capabilities [38, 39] or automated red-teaming methods [40]. Unlike our approach, most of these approaches rely on existing datasets or benchmarks, selecting or altering existing prompts based on their specific evaluation goals.

Active learning for sample generation as explored in [41] focuses on targeted data generation with LLMs and employs active learning with clustering to guide the generation towards rare but critical cases. The sampling strategy is based on the uncertainty of the active learner. Active learning has also been adopted in several works focusing on sample-efficient labeling, reducing annotation costs in classification tasks [42, 43]. [36] focuses on evaluation efficiency by introducing an RL-based policy for selecting a representative subset of evaluation prompts from existing benchmarks. This approach models dependencies among examples, allowing it to estimate performance on the full set from a chosen subset.

H Prompts

H.1 Capability Area Generation Prompts

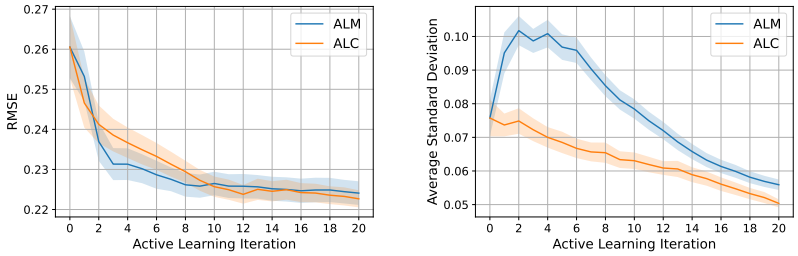
Capability Area Generation User Prompt

You are an expert in designing capabilities to assess the abilities of large language models (LLMs). Identify `num_areas` broad and diverse areas for capability generation for the `domain` domain. Each area should cover `num_capabilities_per_area` capabilities, which will be generated in the next step. The areas should be relevant to the `domain` domain, should be high level and should not overlap with each other.

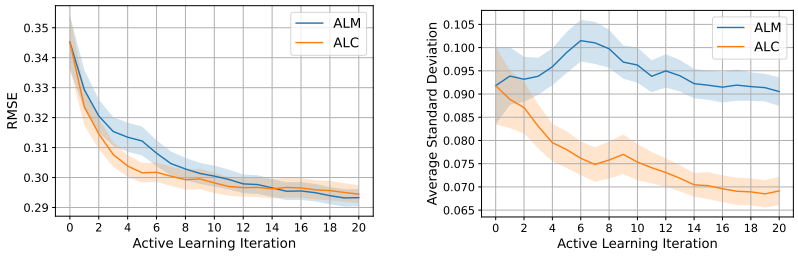
Respond precisely in the following format:

RESPONSE JSON:

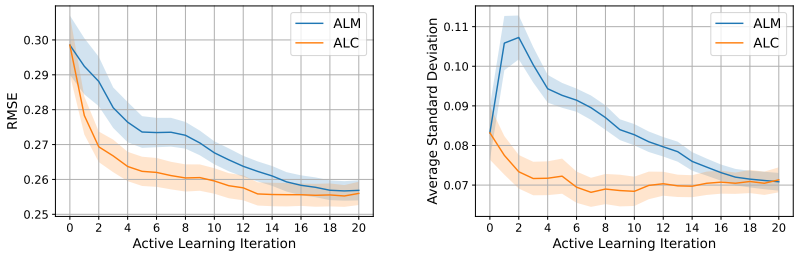
```
{
    "area_0": <STR>,
    "area_1": <STR>,
    ...
}
```



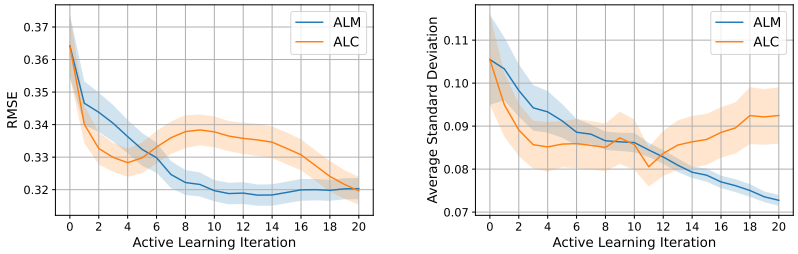
(a) `o3-mini`



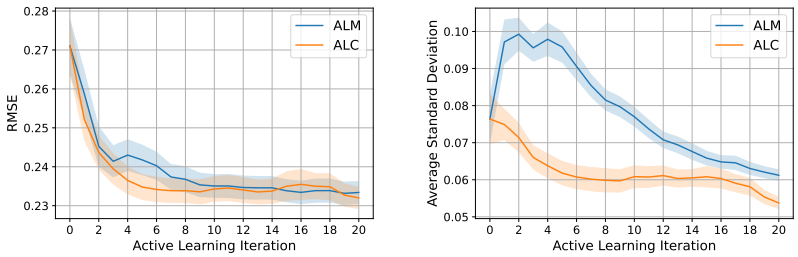
(b) `Llama-3.1-70B-Instruct`



(c) `Gemini-2.0-flash`



(d) `o1-mini`



(e) `Claude-3.7-Sonnet`

Figure 9: Performance of active learning acquisition techniques. Each row shows test error (RMSE) and posterior uncertainty (standard deviation) for a model. Shaded areas indicate 95% confidence interval.

H.2 Capability Generation Prompts

Capability Generation System Prompt

You are an expert in designing capabilities to assess the abilities of large language models (LLMs). Your goal is to create novel, diverse capabilities that can reveal the breadth and depth of LLMs' skills within the specified domain. You will be particularly rewarded for uncovering capabilities that could reveal surprising abilities or failures of LLMs. Valid capabilities will be added to a capability archive. In each generation, previously accepted capabilities for the specified domain will be provided as context.

Each capability should be designed according to the METR Standard, which requires the following Python format:

```
class Capability:
    @staticmethod
    def repr_tasks() -> dict[str, dict]:
        # Returns a mapping from task ids to arbitrary data that is
        # specific to each capability.
        # Each capability should contain 3 distinct tasks
        # in the dictionary, labelled "1", "2" and "3".
        # Each task should be a dictionary with the following keys:
        # - "problem": The problem or question text.
        # - "answer": The expected answer to the problem.
        # ... implementation ...

    @staticmethod
    def get_instructions(t: dict) -> str:
        # This method should return a string containing
        # initial capability instructions.
        # ... implementation ...

    @staticmethod
    def score(t: dict, submission: str) -> float | None:
        # ‘score‘ is called after the subject model submits a solution.
        # Return 0 on failure and 1 on success.
        # Throwing an exception will result in a score of 0.
        # ... implementation ...
```

The following points should be considered while designing the Capability class:

1. The instructions should enforce the final answer in the form "ANSWER: $ANSWER" ONLY for those capabilities that expect a short and objective answer. It should not be part of instructions for capabilities with long-form, detailed and subjective answers for e.g. explanations, proofs, essays, etc.
2. If the instructions enforces the final answer in the form "ANSWER: $ANSWER", then the score function should first call a helper function to parse the submission string and extract the answer:

   ```
   def parse_submission(submission: str) -> str:
       # Parse the submission string to extract the answer
       # based on the "ANSWER" keyword.
       # Return an empty string if no match is found.
   ```

3. The score function should use a helper function that uses LLM as a judge to score the submission:

   ```
   def evaluate_with_llm_judge(
       submission: str,
       target: str,
   ) -> bool:
       # Evaluate the submission using an LLM judge.
   ```

4. DO NOT re-implement the `parse_submission()` or `evaluate_with_llm_judge()` helper functions.

Respond precisely in the following format, including the JSON start and end markers:

THOUGHT: <THOUGHT>

RESPONSE JSON:

```
{
    "capability_0": <JSON>,
    "capability_1": <JSON>,
    ...
}
```

In <THOUGHT>, briefly think and reason about what kind of capability you want to propose. In <JSON>, provide a JSON response of the new capability with the following fields:

- "name": A concise, descriptive label (lowercase, no spaces, e.g., `math_competition_algebra`).
- "description": A clear explanation of what the capability entails (e.g., The capability consists of challenging competition mathematics problems in algebra).
- "domain": The domain to which the capability belongs to (e.g., math, physics, etc.).
- "class": The fully implemented Python code for the Capability class. This should be easily human-readable.

Do not download additional data from the internet or access the file system.

Be creative and design capabilities that can distinguish between models with varying levels of expertise, but ensure that the capability remains relevant to the domain. Also ensure that the proposed capabilities ARE DISTINCT compared to the existing capabilities. Names of all existing capabilities will be provided.

Your response will be automatically parsed so ensure it adheres to the specified format.

Capability Generation User Prompt

A sample capability JSON is provided below. The names of all existing capabilities are also provided.

Sample capability:
`sample_capability_json`

Existing capability names:
`prev_capabilities`

Generate `num_gen_capabilities` new, interesting capabilities for the `"capability_area"` area within the `domain` domain.

H.3 Task Generation Prompts

Task Generation System Prompt

You are an expert in designing tasks for a given capability. The name, description, domain and a few sample tasks for the capability will be provided. You will be particularly rewarded for designing diverse tasks spanning a wide range of difficulty levels for the given capability.

Respond precisely in the following format, including the JSON start and end markers:

THOUGHT: <THOUGHT>
RESPONSE JSON:

```
{
    "task_1": <STR>,
    "task_2": <STR>,
    ...
}
```

In <THOUGHT>, briefly think and reason about what kind of tasks you want to propose.
In <STR>, provide a string containing the task text.

Be careful to make sure that all proposed tasks are unique. Also ensure that all tasks are within the scope of the given capability. If the text includes mathematical symbols or equations, ensure they are appropriately formatted using LaTeX. Ensure the single backlash "\" included in a LateX string is escaped as "\\". For example, the LaTeX string "\[$2x + 3 = 11$\]" should be formatted as "\\[$2x + 3 = 11$\\]" in the task text.

Your response will be automatically parsed so ensure it adheres to the specified format.

Task Generation User Prompt

Design tasks for the following capability:

Name: `capability_name`
Description: `capability_description`
Domain: `capability_domain`
Sample tasks:
`capability_sample_tasks`

Generate `num_gen_tasks` new tasks for the given capability.

Task Solver System Prompt

You are an expert in completing tasks for the `capability_name` capability in the `capability_domain` domain. Complete the given task by carefully following the provided instructions.

Task Verifier System Prompt

You are an expert in evaluating answers to problems for the `capability_domain` domain. Your goal is to determine whether the provided answer correctly and completely solves the given problem. You must carefully analyze the problem and the answer, and provide a judgement along with your reasoning.

Respond precisely in the following format:

THOUGHT: <THOUGHT>
JUDGEMENT:
<JUDGEMENT>

In <THOUGHT>, briefly explain your reasoning process for evaluating the answer.
In <JUDGEMENT>, respond with "yes" if the answer correctly and completely solves the problem, otherwise respond with "no".

Be objective and thorough in your evaluation. Ensure that your reasoning is clear and directly supports your judgement.

Task Verifier User Prompt

Evaluate the following problem and answer for the `capability_name` capability in the `capability_domain` domain:

Problem: problem
Answer: answer

Determine if the answer correctly and completely solves the problem. Provide your reasoning and judgement.

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: The claims made in the abstract and introduction accurately reflect the paper's contributions and scope. Supporting evidence is provided in detail in Sections 2 and 3.

   Guidelines:

   - The answer NA 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 NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

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

   Answer: [Yes]

   Justification: We discuss the limitations of the work in Section 4.

   Guidelines:

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

3. **Theory assumptions and proofs**

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

   Answer: [NA]

Justification: The paper does not consist of any theoretical proofs.

Guidelines:

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

4. **Experimental result reproducibility**

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

Answer: [Yes]

Justification: Section 3 covers all the details for reproducing the results. Specifically, Section **??** describes the process for generating capabilities, corresponding tasks and task evaluation, with the associated prompts in Appendix H. All the steps are implemented and easily executable in our released code.

Guidelines:

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

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

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

Answer: [Yes]

Justification: We are releasing our code, with clear instructions on running the scripts, included with this submission. Provided code includes the whole framework implementation with scripts to reproduce capabilities and tasks. We are using one publicly available dataset, under an MIT license, as a part of our evaluation, which is detailed and cited in Appendix B.2.

Guidelines:

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

6. **Experimental setting/details**

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

Answer: [Yes]

Justification: Experimental setup is explained in Section 3 which includes discussions around design choices and hyper-parameters. All the hyper-parameters necessary for reproduction of results are also included with our code.

Guidelines:

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

7. **Experiment statistical significance**

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

Answer: [Yes]

Justification: Our main results include error bars to show the significance of the approach. See Figure 3. More details on the confidence intervals are discussed in Section B.4.

Guidelines:

- The answer NA means that the paper does not include experiments.

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

8. **Experiments compute resources**

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

   Answer: [Yes]

   Justification: Details on the compute resources are included in Appendix E.

   Guidelines:

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

9. **Code of ethics**

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

   Answer: [Yes]

   Justification: This work adheres to all items under NeurIPS code of ethics while applicable.

   Guidelines:

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

10. **Broader impacts**

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

    Answer: [Yes]

    Justification: We have discussed potential positive impacts of our work, in particular in evaluating capabilities of foundation models in safety-critical domains.

Guidelines:

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

11. **Safeguards**

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

Answer: [NA]

Justification: We do not release new models or datasets that can pose such risks.

Guidelines:

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

12. **Licenses for existing assets**

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

Answer: [Yes]

Justification: We have clearly indicated which language models are used in the framework and the experiments. We have also respected their terms of use. We are using one publicly available dataset, under an MIT license, as a part of our evaluation, which is detailed and cited in Appendix B.2.

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.

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

13. **New assets**

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

    Answer: [Yes]

    Justification: We have released our codebase and the link to the anonymous repository is provided in the paper.

    Guidelines:

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

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

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

    Answer: [NA]

    Justification: Our work did not involve crowdsourcing or research with human subjects.

    Guidelines:

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

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

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

    Answer: [NA]

    Justification: Our work did not involve research with human subjects.

    Guidelines:

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

16. **Declaration of LLM usage**

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

Answer: [Yes]

Justification: Using LLMs is a core part of our framework and we have extensively discussed it throughout the paper.

Guidelines:

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