Abstract:

Benchmarks provide a standardized method for evaluating different AI models, enabling reproducibility and comparison between models, and facilitating scientific progress. As AI models continue to develop rapidly, incorporating new datasets, capabilities, and architectures becomes more complicated. Therefore, the current static benchmarks become increasingly irrelevant. The MLCommons team argues that to make AI benchmarks more relevant, it involves making the benchmarks themselves more dynamic, as well as technical innovations that make it easier for scientists and researchers at all levels to use and contribute to the benchmarks. The current progress in technical innovation is a software that allows for a detailed view of a collection of AI benchmarks to be output in various formats that are easily readable and accessible.

Introduction:

A. What is benchmarking in relation to AI, and how is it used?

AI benchmarking is a method for evaluating the effectiveness of an AI model, such as ChatGPT, using a set of standardized metrics, for example, high school-level math questions. These benchmarks and their results will enable ranking various AI models based on their effectiveness in performing specific tasks. One caveat with these benchmarks is that developers of AI models might focus too much on optimizing their models to perform better on the benchmark itself, rather than on performance when deployed in real-world scenarios. Therefore, it is essential to provide a diverse range of benchmarks to assess the model's performance based on the type of task.

B. Scope

This report focuses on the methods for collecting and validating AI benchmarks, as well as the process of developing software to output the collected information in a readable format. The scope of this work includes researching various benchmarks, testing them, and writing Python code to create software that outputs information about the benchmarks based on an argument from the console. This was done during a 10-week internship, alongside the ML Commons team.

C. Purpose

Our software is designed to make it easier for scientists and researchers to determine which AI models will be most useful for their specific task, based on the data and ratings provided for a wide variety of benchmarks.

This manuscript has been authored by Fermi Forward Discovery Group, LLC under Contract No. 89243024CSC000002 with the U.S. Department of Energy, Office of Science, Office of High Energy Physics.

Methods:

1. Benchmark Data Collection

To begin, we used ChatGPT to compile data on a list of benchmarks provided by the MLCommons team. We then reviewed the output information and wrote it as a YAML file containing metadata for each benchmark, including fields such as benchmark name, task type, evaluation metrics, AI capabilities measured, citation information, and available model results.

2. Benchmark Validation and Standardization

To ensure consistency across all benchmark entries, we developed and applied a validation script that checks each YAML file for required fields and correct formats. The script identifies missing or malformed entries and provides helpful error messages, enabling contributors to easily correct them. This step helps maintain the uniformity across all entries in all the input files.

3. Software Development for Displaying Benchmarks

To make the information easily accessible and readable, we wrote code that converts the YAML format input file to Markdown and LaTeX formats. These formats can be specified using flags in the console, such as “--format .tex”. This script also supports options like filtering columns, combining datasets, and producing standalone or PDF outputs. Both formats included BibTeX citations, which were created and added at the end of the file.

4. Added additional features

We also added additional features, like required and optional fields, author truncation, including BibTeX citations, and checking the format of the input files. These features could be accessed using the console flags that we added documentation for in the GitHub repository.

5. Testing and Iteration

Throughout the development process, we tested the software on multiple real and mock benchmark files to ensure it worked correctly and met the needs of the MLCommons team. Additionally, we have added ratings for every benchmark, using a standard scoring criterion. These ratings were based on the problem specification and constraints, the dataset, performance metrics, the reference solution, and a reproducible protocol. These ratings were created to help contributors understand the effectiveness of each benchmark in the list.

Results:

During the internship, we processed and validated 75 benchmark entries. The Python scripts we created successfully generated the specific file format and also included features that optimized

and made the data easier to read. The command-line prompt features that we implemented are listed below,

- --files / -i
  Specifies the YAML file(s) to be processed. This argument is required and can accept one or more files.

- --format / -f
  Sets the desired output format: either md for Markdown or tex for LaTeX. This is a required argument.

- --outdir / -o
  Indicates the directory where output files should be saved. This allows users to control where their processed files go.

- --authortruncation
  Truncates the number of authors displayed in index or summary tables to keep the output clean and concise. Useful for long author lists.

- --columns
  Let users specify a subset of columns to include in the output, using a comma-separated list (e.g., --columns name,date, domain). This supports custom views.

- --check
  Runs validation checks on the YAML input files to ensure all required fields and formatting rules are met. This mode does not produce an output file.

- --index
  Generates individual pages for each benchmark entry.

- --noratings
  Removes the rating columns from the output file. This is useful if the user wants a simpler view of the benchmark information.

- --required
  When used with --columns, it treats all listed columns as required and checks that they are present in every YAML file.

- --standalone / -s
  For LaTeX output, it includes the complete LaTeX document structure (preamble, document environment), making it ready to compile directly.

- --withcitation
  Adds a BibTeX citation row to the Markdown output. This is particularly helpful for

researchers who need to locate citation information quickly.

We also have a Makefile that facilitates the creation of necessary files more easily, eliminating the need to remember the above flags and the project's directory structure. These command-line prompts allow for multiple YAML files filled with benchmark data to be output in MD and TeX formats with different features to make it easier to read.

Ratings:

We also evaluated benchmarks across five dimensions using a 0–10 scale (0=lowest and 10 = highest).

1. **Problem Specification & Constraints** – Clarity of task, input/output formats, and system constraints (e.g., latency, hardware).

2. **Dataset (FAIR Principles)** – Assessment of findability, accessibility, interoperability, and reusability, including versioning and split structure.

3. **Performance Metrics** – Use of well-defined, quantitative metrics aligned with task goals.

4. **Reference Solution** – Presence and quality of a reproducible baseline or model implementation.

5. **Reproducible Protocol** – Availability of code, environment setup, and instructions to reproduce results.

Each score reflects how completely and transparently the benchmark supports evaluation and replication. We reviewed the documentation and ran these benchmarks on our local Python environment to rate them according to the scale shown above.

Example input and output files are shown below:

One input listing with a few columns shown below (YAML format):

```yaml
-
  - date: '2024-05-01'
    description: The date of availability of the benchmark. If an official release date is not available, use the date of
      adding the entry.
    condition: required
  - version: TODO
    description: The version number of the benchmark
    condition: optional
  - last_updated: 2024-05
    description: 'The date when the entry was last updated. Format: YYYY-mm-dd'
    condition: optional
  - expired: null
    description: An indication if the benchmark is no longer valid.
    condition: optional
  - valid: 'yes'
    description: Identifies if the benchmark is valid at the time of review.
    condition: required
  - name: Jet Classification
    description: The name of the benchmark.
    condition: required
  - url: https://github.com/fastmachinelearning/fastml-science/tree/main/jet-classify
    description: The main URL for this benchmark.
    condition: required
  - doi: TODO
    description: A DOI number that may be associated with the benchmark.
    condition: optional
  - domain: Particle Physics
    description: The scientific domain this benchmark belongs to.
    condition: required
  - focus: Real-time classification of particle jets using HL-LHC simulation features
    description: Short summary of the focus of this benchmark.
    condition: required
  - keywords:
    - classification
    - real-time ML
    - jet tagging
    - QKeras
    description: List of keywords relevant for the benchmark.
    condition: '>=1'
```

Output MD format (3 listings with a few columns shown, and one citation in the footnotes is shown below):

date	name	domain	focus	keywords	task_types	metrics
2024-05-01	Jet Classification	Particle Physics	Real-time classification of particle jets using HL-LHC simulation features	classification, real-time ML, jet tagging, QKeras	Classification	Accuracy, AUC
2024-05-01	Irregular Sensor Data Compression	Particle Physics	Real-time compression of sparse sensor data with autoencoders	compression, autoencoder, sparse data, irregular sampling	Compression	MSE, Compression ratio
2024-05-01	Beam Control	Accelerators and Magnets	Reinforcement learning control of accelerator beam position	RL, beam stabilization, control systems, simulation	Control	Stability, Control loss

1. @article{hawks2022fastml, title={Fast Machine Learning for Science: Benchmarks and Dataset}, author={Hawks, Ben and Tran, Nhan and others}, year={2022}, url={https://arxiv.org/abs/2207.07958} } ↩

Output Latex format in PDF view (11 listings shown and references are below):

Date	Name	Domain	Focus	Task Types	Metrics	Models	Citation
2020-09-07	MMLU (Massive Multitask Language Understanding)	Multidomain	Academic knowledge and reasoning across 57 subjects	Multiple choice	Accuracy	GPT-4o, Gemini 1.5 Pro, o1, DeepSeek-R1	[1] ⇒
2023-11-20	GPQA Diamond	Science	Graduate-level scientific reasoning	Multiple choice, Multi-step QA	Accuracy	o1, DeepSeek-R1	[2] ⇒
2018-03-14	ARC-Challenge (Advanced Reasoning Challenge)	Science	Grade-school science with reasoning emphasis	Multiple choice	Accuracy	GPT-4, Claude	[3] ⇒
2025-01-24	Humanity's Last Exam	Multidomain	Broad cross-domain academic reasoning	Multiple choice	Accuracy		[4] ⇒
2024-11-07	FrontierMath	Mathematics	Challenging advanced mathematical reasoning	Problem solving	Accuracy		[5] ⇒
2024-07-18	SciCode	Scientific Programming	Scientific code generation and problem solving	Coding	Solve rate (percent)	Claude3.5-Sonnet	[6] ⇒
2025-03-13	AIME (American Invitational Mathematics Examination)	Mathematics	Pre-college advanced problem solving	Problem solving	Accuracy		[7] ⇒
2025-02-15	MATH-500	Mathematics	Math reasoning generalization	Problem solving	Accuracy		[8] ⇒
2024-04-02	CURIE (Scientific Long-Context Understanding, Reasoning and Information Extraction)	Multidomain Science	Long-context scientific reasoning	Information extraction, Reasoning, Concept tracking, Aggregation, Algebraic manipulation, Multimodal comprehension	Accuracy		[9] ⇒
2023-01-26	FEABench (Finite Element Analysis Benchmark)	Computational Engineering	FEA simulation accuracy and performance	Simulation, Performance evaluation	Solve time, Error norm	FEniCS, deal.II	[10] ⇒
2024-07-12	SPIQA (Scientific Paper Image Question Answering)	Computer Science	Multimodal QA on scientific figures	Question answering, Multimodal QA, Chain-of-Thought evaluation	Accuracy, F1 score	Chain-of-Thought models, Multimodal QA systems	[11] ⇒

References

[1] D. Hendrycks, C. Burns, S. Kadavath, *et al.*, "Measuring massive multitask language understanding," *arXiv preprint arXiv:2009.03300*, 2021. [Online]. Available: https://arxiv.org/abs/2009.03300.

[2] D. Rein, B. L. Hou, A. C. Stickland, *et al.*, *Gpqa: A graduate-level google-proof q and a benchmark*, 2023. [Online]. Available: https://arxiv.org/abs/2311.12022.

[3] P. Clark, I. Cowhey, O. Etzioni, *et al.*, "Think you have solved question answering? try arc, the ai2 reasoning challenge," in *EMNLP 2018*, 2018, pp. 237–248. [Online]. Available: https://allenai.org/data/arc.

[4] L. Phan, A. Gatti, Z. Han, *et al.*, *Humanity's last exam*, 2025. [Online]. Available: https://arxiv.org/abs/2501.14249.

[5] E. Glazer, E. Erdil, T. Besiroglu, *et al.*, *Frontiermath: A benchmark for evaluating advanced mathematical reasoning in ai*, 2024. [Online]. Available: https://arxiv.org/abs/2411.04872.

[6] M. Tian, L. Gao, S. Zhang, *et al.*, *Scicode: A research coding benchmark curated by scientists*, 2024. [Online]. Available: https://arxiv.org/abs/2407.13168.

[7] TBD, *Aime*, [Online accessed 2025-06-24], Mar. 2025. [Online]. Available: https://www.vals.ai/benchmarks/aime-2025-03-13.

[8] HuggingFaceH4, *Math-500*, 2025. [Online]. Available: https://huggingface.co/datasets/HuggingFaceH4/MATH-500.

[9] T. A. authors, *Scientific reasoning benchmarks from the curie dataset*, 2024. [Online]. Available: https://arxiv.org/abs/2404.02029.

[10] A. Institute, *Feabench: A finite element analysis benchmark*, 2023. [Online]. Available: https://github.com/alleninstitute/feabench.

[11] X. Zhong, Y. Gao, and S. Gururangan, "Spiqa: Scientific paper image question answering," 2024. [Online]. Available: https://arxiv.org/abs/2407.09413.

[12] D. Jin, Y. Li, Y. Zhang, *et al.*, "What disease does this patient have? a large-scale open-domain question answering dataset from medical exams," 2020. [Online]. Available: https://arxiv.org/abs/2009.13081.

Ratings are shown as radar plots in the output for both Markdown and LaTeX formats:

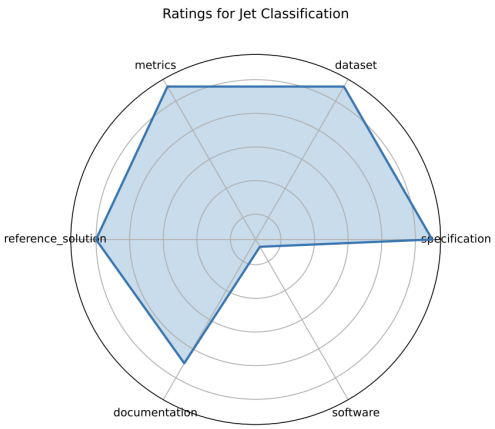

Figure 1: Jet Classification

Radar Plots and Citations:

Radar plots provide an at-a-glance comparison of each benchmark's strengths and weaknesses. For instance, benchmarks with strong datasets but weak reproducibility protocols are immediately identifiable by their skewed plot shapes. In Markdown, each benchmark is also accompanied by footnote citations and links. While in LaTeX, each benchmark is also accompanied by references and links.

Evaluation:

Based on the features we added, we successfully output the proper information for all 75 benchmarks, including a rating. We were also able to begin testing these benchmarks to ensure they work on a standard Python environment, such as Conda. Based on the feedback we received on our software, the software that we developed was able to output the data from each benchmark in an easy-to-read format.

Design tradeoffs and justification:

We decided to use LaTeX and Markdown files instead of the more popular JSON format because JSON does not allow multiline text, reading JSON is more difficult as it includes { }, no comments are allowed in the file, and the syntax of YAML is cleaner/more straightforward. So, we decided to use these two formats as the possible output formats that can be specified by the user.

Potential limitations and future goals:

The main limitation of this project is that we were unable to thoroughly test every benchmark to make sure they work on our local Python environment due to time constraints. This would be something we could do in the future to ensure that software users are assured that each benchmark contains all the necessary information. Several entries also required additional metadata from their sources. These were flagged with TODOs in the YAML and will be prioritized for future validation.

Conclusion:

Overall, this project provided a valuable opportunity to explore the intersection of AI evaluation, reproducible research, and software engineering. By building tools that simplify benchmarking, we contribute to the broader goal of making AI research more transparent, accessible, and impactful.

Acknowledgments:

- **This manuscript has been authored by Fermi Research Alliance, LLC under Contract No. DE-AC02-07CH11359 with the U.S. Department of Energy, Office of Science, Office of High Energy Physics.**
- **This work was supported in part by the U.S. Department of Energy, Office of Science, Office of Workforce Development for Teachers and Scientists (WDTS) under the Community College Internship (CCI)**