Scientific machine learning benchmarks

Jeyan Thiyagalingam, Mallikarjun Shankar, Geoffrey Fox and Tony Hey

Abstract | Deep learning has transformed the use of machine learning technologies for the analysis of large experimental datasets. In science, such datasets are typically generated by large-scale experimental facilities, and machine learning focuses on the identification of patterns, trends and anomalies to extract meaningful scientific insights from the data. In upcoming experimental facilities, such as the Extreme Photonics Application Centre (EPAC) in the UK or the international Square Kilometre Array (SKA), the rate of data generation and the scale of data volumes will increasingly require the use of more automated data analysis. However, at present, identifying the most appropriate machine learning algorithm for the analysis of any given scientific dataset is a challenge due to the potential applicability of many different machine learning frameworks, computer architectures and machine learning models. Historically, for modelling and simulation on high-performance computing systems, these issues have been addressed through benchmarking computer applications, algorithms and architectures. Extending such a benchmarking approach and identifying metrics for the application of machine learning methods to open, curated scientific datasets is a new challenge for both scientists and computer scientists. Here, we introduce the concept of machine learning benchmarks for science and review existing approaches. As an example, we describe the SciMLBench suite of scientific machine learning benchmarks.

In the past decade, a subfield of artificial intelligence (AI), namely, deep learning (DL) neural networks (or deep neural networks, DNNs), has enabled significant breakthroughs in many scientifically and commercially important applications[1]. Such neural networks are themselves a subset of a wide range of machine learning (ML) methods.

ML methods have been widely used for many years in several domains of science, but DNNs have been transformational and are gaining a lot of traction in many scientific communities[2,3]. Most of the national, international and big laboratories that host large-scale experimental facilities, as well as commercial entities capable of large-scale data processing (big tech), are now relying on DNN-based data analytic methods to extract insights from their increasingly large datasets. A recent success from industry is the use of DL to find solutions to the protein folding problem[4]. Current developments point towards specializing these ML approaches to be more domain-specific and domain-aware[5–7], and aiming to connect the apparent 'black-box' successes of DNNs with the well-understood approaches from science.

The overarching scope of ML in science is broad. A non-exhaustive list includes the identification of patterns, anomalies and trends from relevant scientific datasets, the classification and prediction of such patterns and the clustering of data. The data are not always experimental or observational but can also be synthetic data. There are three approaches for developing ML-based solutions, namely, supervised, unsupervised and reinforcement learning. In supervised learning, the ML model is trained with examples to perform a given task. In this case, the training data used must contain the 'ground truth' or labels. Supervised learning is, therefore, possible only when there is a labelled subset of the data. Once trained, the learned model can be deployed for real-time usage, such as pattern classification or estimation — which is often referred to as 'inference'. Because of the difficulty in generating labelled data for supervised learning, particularly for experimental datasets, it is often difficult to apply supervised learning directly. To circumvent this limitation, training is often performed on simulated data, which provides an opportunity to have relevant labels. However, the simulated data may not be representative of the real data and the model may, therefore, not perform satisfactorily when used for inferencing. The unsupervised learning technique, in contrast, does not rely on labels. A simple example of this technique is clustering, where the aim is to identify several groups of data points that have common features. Another example is identification of anomalies in data. Example algorithms include k-means clustering[8], Support Vector Machines (SVMs)[9] or neural-network-based autoencoders[10]. Finally, reinforcement learning relies on a trial-and-error approach to learn a given task, with the learning system being positively rewarded whenever it behaves correctly and penalized whenever it behaves incorrectly[11]. Each of these learning paradigms has a large number of algorithms, and modern developmental approaches are often hybrid and use one or more of these techniques together. This leaves many choices of ML algorithms for any given problem.

In practice, the selection of an ML algorithm for a given scientific problem is more complex than just selecting one of the ML technologies and any particular algorithm. The selection of the most effective ML algorithm is based on many factors, including the type, quantity and quality of the training data, the availability of labelled data, the type of problem being addressed (prediction, classification and so on), the overall accuracy and performance required, and the hardware systems available for training and inferencing. With such a multidimensional problem consisting of a choice of ML algorithms, hardware architectures and a range of scientific problems, selecting an optimal

ML algorithm for a given task is not trivial. This constitutes a significant barrier for many scientists wishing to use modern ML methods in their scientific research.

In this Perspective, we discuss what are suitable scientific ML benchmarks and how to develop guidelines and best practices to assist the scientific community in successfully exploiting these methods. Developing such guidelines and best practices at the community level will not only benefit the science community but also highlight where further research into ML algorithms, computer architectures and software solutions for using ML in scientific applications is needed.

We refer to the development of guidelines and best practices as benchmarking. The applications used to demonstrate the guideline and best practices are referred to as benchmarks. The notion of benchmarking computer systems and applications has been a fundamental cornerstone of computer science, particularly for compiler, architectural and system development, with a key focus on using benchmarks for ranking systems, such as the TOP500 or Green500 (REFS[12–16]). However, our notion of scientific ML benchmarking has a different focus and, in this Perspective, we restrict the term 'benchmarking' to ML techniques applied to scientific datasets. Firstly, these ML benchmarks can be considered as blueprints for use on a range of scientific problems, and, hence, are aimed at fostering the use of ML in science more generally. Secondly, by using these ML benchmarks, a number of aspects in an ML ecosystem can be compared and contrasted. For example, it is possible to rank different computer architectures for their performance or to rank different ML algorithms for their effectiveness. Thirdly, these ML benchmarks are accompanied by relevant scientific datasets on which the training and/or inference will be based. This is different to conventional benchmarks for high-performance computing (HPC), where there is little dependency on datasets. The establishment of a set of open, curated scientific datasets with associated ML benchmarks is, therefore, an important step for scientists to be able to effectively use ML methods in their research and also to identify further directions for ML research.

Machine learning benchmarks for science

In this section, we discuss the elements of a scientific benchmark and the focus of scientific benchmarking, along with relevant examples.

Elements of a benchmark for science. As discussed above, a scientific ML benchmark is underpinned by a scientific problem and should have two elements: first, the dataset on which this benchmark is trained or inferenced upon and, second, a reference implementation, which can be in any programming language (such as Python or C++). The scientific problem can be from any scientific domain. A collection of such benchmarks can make up a benchmark suite, as illustrated in FIG. 1.

Focus of benchmarking. There are three separate aspects of scientific benchmarking that apply in the context of ML benchmarks for science, namely, scientific ML benchmarking, application benchmarking and system benchmarking. These are explained below.

- Scientific ML benchmarking. This is concerned with algorithmic improvements that help reach the scientific targets specified for a given dataset. In this situation, one wishes to test algorithms and their performance on fixed data assets, typically with the same underlying hardware and software environment. This type of benchmark is characterized by the dataset, together with some specific scientific objectives. The data are obtained from a scientific experiment and should be rich enough to allow different methods of analysis and exploration. Examples of metrics could include the F1 score for training accuracy, time to solution and any domain-specific metric(s). A more detailed discussion on metrics can be found in the next section.
- Application benchmarking. This aspect of ML benchmarks is concerned with exploring the performance of the complete ML application (covering loading of inputs from files, pre-processing, application of ML, post-processing and writing outputs to files) on different hardware and software environments. This can also be referred to as an end-to-end ML application benchmark. A typical performance target for these types of benchmarks may include training time or even complete time to solution. Such application benchmarks can also be used to evaluate the performance of the overall system, as well as that of particular subsystems (hardware, software libraries, runtime environments, file systems and so on). For example, in the case of image classification, the relevant performance metric could be a throughput measure (for example, images per second) for training or inference, or the time to solution of the classification problem (including I/O, ML, and pre-processing and post-processing), or the scaling properties of the application.
- System benchmarking. This is concerned with investigating performance effects of the system hardware architecture on improving the scientific outcomes/targets. These benchmarks have similarities with application benchmarks, but they are characterized by primarily focusing on a specific operation that exercises a particular part of the system, independent of the broader system environment. Suitable metrics could be time to solution, the number of floating-point operations per second achieved or aspects of network and data movement performance.

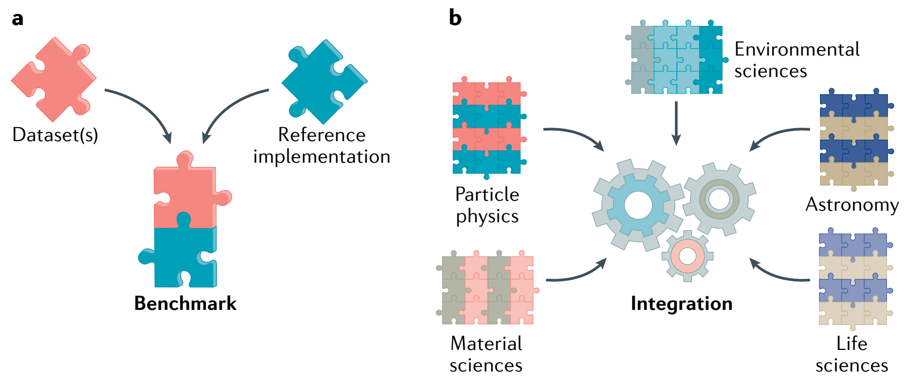

Fig. 1 | **The notion of a machine learning benchmark and a benchmark suite. a** | Elements of a scientific machine learning (ML) benchmark. **b** | Building a scientific ML benchmark suite that integrates different scientific ML benchmarks from various scientific disciplines.

Examples of scientific machine learning benchmarks. Scientific ML benchmarks are ML applications that solve a particular scientific problem from a specific scientific domain. For example, this can be as simple as an application that classifies the experimental data in some way, or as complex as inferring the properties of a material from neutron scattering data. Some examples are given below.

- Inferring the structure of multiphase materials from X-ray diffuse multiple scattering data. Here, ML is used to automatically identify the phases of materials using classification[2].
- Estimating the photometric red shifts of galaxies from survey data[17]. Here, ML is used for estimation.
- Clustering of microcracks in a material using X-ray scattering data[18]. Here, ML uses an unsupervised learning technique.
- Removing noise from microscope data to improve the quality of images. ML is used for its capability to perform high-quality regression of pixel values[19].

More detailed examples are provided in later sections.

The benchmarking process

Although it is possible to provide a collection of ML-specific scientific applications (with relevant datasets) as benchmarks for any of the purposes mentioned above, the exact process of benchmarking requires the following elements, given below.

- Metrics of choice. First, depending on the focus, the exact metric by which different benchmarks are compared may vary. For example, if science is the focus, then this metric may vary from benchmark to benchmark. However, if the focus is system-level benchmarking, it is possible to agree on a common set of metrics that can span across a range of applications. However, in the context of ML, owing to the uncertainty around the underlying ML model(s), dataset(s) and system hardware (for example mixed-precision systems), it may be more meaningful to ensure that uncertainties of the benchmark outputs are quantified and compared wherever necessary. Likewise, the level of explainability of methods (and, hence, outputs) can be a differentiator between different ML methods and, hence, of benchmarks. In this way, the explainability of different ML implementations for a given benchmark problem could be considered as a metric as well, provided this can be well quantified. Another axis could be around energy efficiency, such as the ability of an ML implementation to perform training or inference with minimum power or energy requirements. It is clearly essential to agree upon the appropriate figures of merit and metrics to be used for comparing different implementations of benchmarks.
- Framework. Providing just a collection of disparate applications without a coherent mechanism for evaluation requires users to perform a set of fairly complex benchmarking operations that are relevant to their specific goals. Ideally, the benchmark suite should, therefore, offer a framework that not only helps users to achieve their specific goals but also unifies aspects that are common to all applications in the suite, such as benchmark portability, flexibility and logging.
- Reporting and compliance. Finally, how these results are reported is important. In many cases, a benchmark framework as discussed above addresses this concern. However, there are often some specific compliance aspects that must be followed to ensure that the benchmarking process is carried out fairly across different hardware platforms.

There are also a number of challenges that need to be addressed when dealing with the development of ML benchmarks; these are given below.

- Data. In the previous section, we highlighted the significance of data when using ML for scientific problems. The availability of curated, large-scale, scientific datasets — which can be either experimental or simulated data — is the key to developing useful ML benchmarks for science. Although a lot of scientific data are openly available, the curation, maintenance and distribution of large-scale datasets for public consumption is a challenging process. A good benchmarking suite needs to provide a wide range of curated scientific datasets coupled with the relevant applications. Reliance on external datasets has the danger of not having full control or even access to those datasets.
- Distribution. A scientific ML benchmark comprises a reference ML implementation together with a relevant dataset, and both of these must be available to the users. Since realistic dataset sizes can be in the terabytes range, the access and downloading of these datasets is not always straightforward.
- Coverage. Benchmarking is a very broad topic and providing benchmarks to cover the different focus areas highlighted above, across a range of scientific disciplines, is not a trivial task. A good benchmark suite should provide a good coverage of methods and goals, and should be extensible.
- Extensibility. Although developing scientific ML benchmarks can be valuable for scientists, it can be time consuming to develop benchmarking-specific codes. If the original scientific application needs substantial refactoring to be converted into a benchmark, this will not be an attractive option for scientists. Any benchmarking framework should, therefore, try to minimize the amount of code refactoring required for conversion into a benchmark.

In addition to these challenges, ML benchmarks need to address a number of other issues, such as problems with overtraining and overfitting. In most cases, such issues can be covered by requiring compliance with some general rules for the benchmarks — such as specifying the set of hyperparameters that are open to tuning. Although one may consider these as aspects of scientific ML benchmarking, they are best handled through explicit specification of the rules of the benchmarking process. For example, the training and validation data, and cross-validation procedures, should aim to mitigate the dangers of overfitting.

Benchmarking initiatives

Comparing different ML techniques is not a new requirement and is increasingly becoming common in ML research. In fact, this approach has been fundamental for the development of various ML techniques. For example, the ImageNet[20,21] dataset spurred a competition to improve computer image analysis and understanding, and has been widely recognized for driving innovation in DL. A recent example of an application and system benchmark is the High-Performance LINPACK for Accelerator Introspection (HPL-AI) benchmark[22], which aims to drive AI innovation by focusing on the performance benefits of reduced (and mixed) precision computing. However, providing a blueprint of applications, guidelines and best practices in the context of scientific ML is a relatively new and unaddressed requirement. There have been a number of efforts on this aspect that address some of the challenges we highlighted above. In this brief overview of these benchmarking initiatives, we explicitly exclude conventional benchmarking activities in other areas of computer science, such as benchmarks for HPC systems, compilers and subsystems, such as memory, storage and networking[12,23].

Instead of giving an exhaustive technical review covering very-fine-grained aspects, we give a high-level overview of the various ML benchmark initiatives, focusing on the requirements discussed in the previous sections. We shall, therefore, cover the following aspects:

- Benchmark focus: science, application (end-to-end) and system.
- Benchmark process: metrics, framework, reporting and compliance.
- Benchmark challenges: data, distribution, coverage and extensibility.

In the context of ML benchmarking, there are several initiatives, such as Deep500 (REF.[24]), RLBench[25], CORAL-2 (REF.[26]), DAWNBench[27], AIBench[28], MLCommons[29] and SciMLBench[30], as well as specific community initiatives (such as the well-known community competitions organized by Kaggle[31]). We overview these initiatives below and note that a specific benchmarking initiative may or may not support all the aspects listed above or, in some cases, may only offer partial support.

Deep500. The Deep500 (REF.[24]) initiative proposes a customizable and modular software infrastructure to aid in comparing the wide range of DL frameworks, algorithms, libraries and techniques. The key idea behind Deep500 is its modular design, where DL is factorized into four distinct levels: operators, network processing, training and distributed training. Although this approach aims to be neutral and overarching, and also able to accommodate a wide variety of techniques and methods, the process of mapping a code to a new framework has impeded its adoption for new benchmark development. Furthermore, despite its key focus on DL, neural networks and a very customizable framework, benchmarks or applications are not included by default and are left for the end user to provide, as is support for reporting. The main limitation is the lack of a suite of representative benchmarks.

RLBench. RLBench[25] is a benchmark and learning environment featuring hundreds of unique, hand-crafted tasks. The focus is on a set of tasks to evaluate new algorithmic developments around reinforcement learning, imitation learning, multitask learning, geometric computer vision and, in particular, few-shot learning. The tasks are very specific and can be considered as building blocks of large-scale applications. However, the environment currently lacks support for the classes of benchmarking discussed above.

CORAL-2. The CORAL-2 (REF.[26]) benchmarks are computational problems relevant to a scientific domain or to data science, and are typically backed by a community code. Vendors are then expected to evaluate and optimize these codes to demonstrate the value of their proposed hardware in accelerating computational science. This allows a vendor to rigorously demonstrate the performance capabilities and characteristics of a proposed machine on a benchmark suite that should be relevant for computational scientists. The ML and data science tools in CORAL-2 include a number of ML techniques across two suites, namely, the big data analytics (BDAS) and DL (DLS) suites. Whereas the BDAS suite covers conventional ML techniques, such as principal components analysis (PCA), k-means clustering and SVMs, the DLS suite relies on the ImageNet[20,21] and CANDLE[32] benchmarks, which are primarily used for testing scalability aspects, rather than purely focusing on the science. Similarly, the BDAS suite aims to exercise the memory constraints (PCA), computing capabilities (SVMs) and/or both these aspects (k-means) and is also concerned with communication characteristics. Although these benchmarks are oriented at ML, the constraints and benchmark targets are narrowly specified and emphasize scalability capabilities. The overall coverage of science in the CORAL-2 benchmark suite is quite broad, but the footprint of the ML techniques is limited to the BDAS and DLS suites, and there is little focus on scientific data distribution for algorithm improvement.

AIBench. The AIBench initiative is supported by the International Open Benchmark Council (BenchCouncil)[28]. The Council is a non-profit international organization that aims to promote standardizing, benchmarking, evaluating and incubating big data, AI and other emerging technologies. The scope of AIBench is very comprehensive and includes a broad range of internet services, including search engines, social networks and e-commerce. The underlying ML-specific tasks in these areas include image classification, image generation, translation (image-to-text, image-to-image, text-to-image, text-to-text), object detection, text summarization, advertising and natural language processing. The relevant datasets are open and the primary metric is system performance for a fixed target. One of the important components of the AIBench initiative is HPC AI500 (REF.[33]), a standalone benchmark suite for evaluating HPC systems running DL workloads. The suite covers a number of representative scientific problems from various domains, with each workload being a real-world scientific DL application, such as extreme weather analysis[33]. The suite includes reference implementations, datasets and other relevant software, along with relevant metrics. This HPC ML suite compares best to the SciMLBench work discussed below. The AIBench environment also enforces some level of compliance for reporting ranking information of hardware systems.

DAWNBench. DAWNBench[27] is a benchmark suite for end-to-end DL training and inference. The end-to-end aspect is ideal for application-level and system-level benchmarking. Instead of focusing on model accuracy, DAWNBench provides common DL workloads for quantifying training time, training cost, inference latency and inference cost across different optimization strategies, model architectures, software frameworks, clouds and hardware. There are two key benchmarks in the suite — image classification (using the ImageNet and CIFAR-10 (REF.[34]) datasets) and natural-language-processing-based question answering[35] (based on the Stanford Question Answering Dataset or SQuAD[35]) that covers both training and inference. DAWNBench does not offer the notion of a framework and does not have a focus on science. With key metrics around time and cost (for training and inference), DAWNBench is predominantly targeted towards end-to-end system and application performance. Although the datasets are public and open, no distribution mechanisms have been adopted by DAWNBench.

Benchmarks from the MLCommons working groups. MLCommons is an international initiative aimed at improving all aspects of the ML landscape and covers benchmarking, datasets and best practices. The consortium has several working groups with different foci for ML applications. Among these working groups, two are of interest here: HPC and Science. The MLCommons HPC benchmark[29] suite focuses on scientific applications that use ML, and especially DL, at the HPC scale. The codes and data are specified in such a way that execution of the benchmarks on supercomputers will help understand detailed aspects of system performance. The focus is on performance characteristics particularly relevant to HPC applications, such as model–system interactions, optimization of the workload execution and reducing execution and throughput bottlenecks. The HPC orientation also drives this effort towards exploration of benchmark scalability.

By contrast, the MLCommons Science benchmark[36] suite focuses specifically on the application of ML methods to scientific applications and includes application examples from several scientific domains. The recently announced information on the science benchmarks at Supercomputing 2021 will spur improvements in defining datasets for advancing ML for science. The suite currently lacks a supportive framework for running the benchmarks but, as with the rest of MLCommons, does enforce compliance for reporting of the results. The benchmarks cover the three areas of benchmarking — science, application and system.

SciMLBench. The Scientific Machine Learning Benchmark suite — or SciMLBench[30] — is specifically focused on scientific ML and covers nearly every aspect of the cases discussed in the previous sections. A detailed description of the SciMLBench initiative is described in the next section.

Other community initiatives. In addition to various efforts mentioned above, there are other efforts towards AI benchmarking by specific research communities. Two examples are WeatherBench[37] and MAELSTROM[38] from the weather and climate communities, both of which have specific goals and include relevant data and baseline techniques. However, these efforts are not full benchmark suites, and, instead, are engineered as individual benchmarks, ideally to be integrated as part of a suite.

Although community-based competitions, such as Kaggle[31], can be seen as a benchmarking activity, these competitions do not have a coherent methodology or a controlled approach for developing benchmarks. In particular, the competitions do not provide a framework for running the benchmarks, nor do they consider data distribution methods. Each competition is individually constructed and relies on its own dataset, set of rules and compliance metrics. The competitions address concerns such as dataset curation, choice of metric, presentation of results and robustness against overfitting, for example. Although such challenge competitions can provide a blueprint for using ML technologies for specific research communities, the competitions are generally short lived and are, therefore, unlikely to deliver best practices or guidelines for the long term.

The SciMLBench approach

The SciMLBench approach has been developed by the authors of this article, members of the Scientific Machine Learning Group at the Rutherford Appleton Laboratory, in collaboration with researchers at Oak Ridge National Laboratory and at the University of Virginia. Among all the approaches reviewed above, only the SciMLBench benchmark suite attempts to address all of the concerns discussed previously. To the best of our knowledge, the SciMLBench approach is unique in its versatility compared with the other approaches and its key focus is on scientific ML.

Core components. SciMLBench has three components, given below.

- Benchmarks. The benchmarks are ML applications written in Python that perform a specific scientific task. These applications are included by default and users are not required to find or write their own applications. On the scale of micro-apps, mini-apps and apps, these codes are full-fledged applications. Each benchmark aims to solve a specific scientific problem (such as those discussed earlier). The set of benchmarks are organized into specific themes, including DL-focused benchmarks, training or inference-intensive benchmarks, benchmarks emphasizing uncertainty quantification, benchmarks focusing on specific scientific problems (such as denoising[19], nonlinear dynamical systems[5], and physics-informed neural networks[5]) and benchmarks focusing on surrogate modelling[39]. Although the current set of benchmarks and their relevant datasets are all image based, the design of SciMLBench allows for datasets that are multimodal or include mixed types of data.
- Datasets. Each benchmark relies on one or more datasets that can be used, for example, for training and/or inferencing. These datasets are open, task or domain specific and compliant with respect to the FAIR guidelines (Findable, Accessible, Interoperable and Reusable[40]). Since most of these datasets are large, they are hosted separately on one of the laboratory servers (or mirrors) and are automatically or explicitly downloaded on demand.
- Framework. The framework serves two purposes. Firstly, at the user level, it facilitates an easier approach to the actual benchmarking, logging and reporting of the results. Secondly, at the developer level, it provides a coherent application programming interface (API) for unifying and simplifying the development of ML benchmarks.

The SciML framework is the basic fabric upon which the benchmarks are built. It is both extensible and customizable, and offers a set of APIs. These APIs enable easier development of benchmarks based on this framework and are defined with layers of abstractions. Example APIs (and their abstractions) are given below.

- The entry point for the framework to run the benchmark in training mode, abstracted to all benchmark developers (scientists), requires the API to follow a specific signature. If defined, the benchmark can then be called to run in training mode. If this is undefined and the benchmark is invoked in training mode, it will fail.
- The entry point for the framework to run the benchmark in inference mode, abstracted to all benchmark developers (scientists), requires the API to follow a specific signature. If defined, the benchmark can be called to run in inference mode. If this is undefined and the benchmark is invoked in inference mode, it will fail.
- Control of logging. APIs for logging of details are available at different granularities. At the highest (abstraction) level, this can be simply the starting and stopping of logging. At the fine-grained level, it can be controlling what is specifically being logged.
- Controlling the execution of benchmarks. These APIs are designed for advanced benchmark developers to control aspects around the actual execution of benchmarks and would be expected to be seldom used by scientists.

These APIs, in contrast to APIs from other frameworks, such as Deep500, are layered and are not fine grained. In other words, APIs from SciMLBench are abstracted enough for the benchmarking process to be automated as much as possible, instead of providing APIs for obtaining fine-grained measurements, such as runtime or I/O or communication times. In fact, SciMLBench retains these measurements and makes them available for detailed analysis, but the focus is on science rather than on performance. In addition, these APIs are totally independent of the application, whereas APIs in frameworks like Deep500 are intended to reflect the operational semantics of the layers or operations of the neural networks.

The SciMLBench framework is independent of architecture, and the minimum system requirement is determined by the specific benchmark. There is a built-in logging mechanism that captures all potential system-level and benchmark-level outputs during execution, leaving end users or benchmark designers to decide the content and format of the report from these detailed logs. The central component that links benchmarks, datasets and the framework is the framework configuration tool. The most attractive part of the framework is the possibility of simply using existing codes as benchmarks, with only a few API calls necessary to register the benchmarks. Finally, the framework is designed with scalability in mind, so that benchmarks can be run on any computer, ranging from a single system to a large-scale supercomputer. This level of support is essential, even if the included benchmarks, in their own, are scalable.

Benchmarks and datasets. The currently released version of SciMLBench has three benchmarks with their associated datasets. The benchmarks from this release represent scientific problems drawn from material sciences and environmental sciences, listed below.

- Diffuse multiple scattering (DMS_Structure). This benchmark uses ML for classifying the structure of multiphase materials from X-ray scattering patterns. More specifically, the ML-based approach enables automatic identification of phases. This application is particularly useful for the materials science community, as diffuse multiple scattering allows investigation of multiphase materials from a single measurement — something that is not possible with standard X-ray experiments. However, manual analysis of the data can be extremely laborious, involving searching for patterns to identify important motifs (triple intersections) that allow for inference of information. This is a multilabel classification problem (as opposed to a binary classification problem, as in the cloud masking example discussed below). The benchmark relies on a simulated dataset of size 8.6 GB with three-channel images of resolution 487 × 195 pixels.
- Cloud masking (SLSTR_Cloud). Given a set of satellite images, the challenge for this benchmark is to classify each pixel of each satellite image as either cloud or non-cloud (clear sky). This problem is known as 'cloud masking' and is crucial for several important applications in earth observation. In a conventional, non-ML setting, this task is typically performed using either thresholding or Bayesian methods. The benchmark exercises DL and includes two datasets, DS1-Cloud and DS2-Cloud, with sizes of 180 GB and 1.2 TB, respectively. The datasets contain multispectral images with resolutions of 2,400 × 3,000 pixels and 1,200 × 1,500 pixels.
- Electron microscopy image denoising (EM_Denoise). This benchmark uses ML for removing noise from electron microscopy images. This improves the signal-to-noise ratio of the image and is often used as a precursor to more complex techniques, such as surface reconstruction or tomographic projections. Effective denoising can facilitate low-dose experiments in producing images with a quality comparable with that obtained in high-dose experiments. Likewise, greater time resolution can also be achieved with the aid of effective image denoising procedures. This benchmark exercises complex DL techniques on a simulated dataset of size 5 GB, consisting of 256 × 256 images covering noised and denoised (ground truth) datasets.

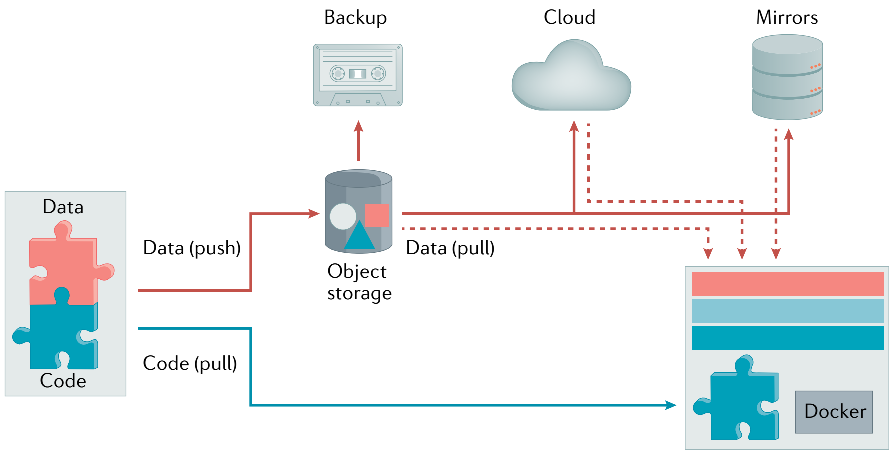

Fig. 2 | **Moving the benchmark datasets to the evaluation point.** A benchmark has two components: a code and the associated datasets. Whenever a user wants to use a benchmark, the code component can easily be directly downloaded from the server. The data component, however, requires careful delivery. The associated datasets are often too large for it to be possible to download them from the server through direct download. Instead, they are pushed to the object storage, where they are carefully curated and backed up. This curated dataset is then pulled on demand by the user when a benchmark that requires this dataset is to be used. Because the exact location of the dataset can lead to delays, these datasets are often mirrored and can also be made available as part of cloud environments. This way, the download location can be opted for by the user (or automatically selected by the downloading component). The dotted lines imply that the data can come from any of the locations and can be specified. The 'pull' aspect means that the data are downloaded on demand (pulled by the user). The 'push' component means that the dataset distribution is managed by a server or the framework.

The next release of the suite will include several more examples from various domains with large datasets, such as a scanning electron tomography benchmark from material sciences, a benchmark for quantifying damage to optical lenses in laser physics and another denoising benchmark for cryogenic electron microscopic images from the life sciences domain.

Benchmark focus. With the full-fledged capability of the framework to log all activities, and with a detailed set of metrics, it is possible for the framework to collect a wide range of performance details that can later be used for deciding the focus. For example, SciMLBench can be used for science benchmarking (to improve scientific results through different ML approaches), application-level benchmarking and system-level benchmarking (gathering end-to-end performance, including I/O and network performance). This is made possible thanks to the detailed logging mechanisms within the framework. These logging mechanisms rely on various low-level details for gathering system-specific aspects, such as memory, GPU or CPU usages. Furthermore, there are APIs available for logging all the way from the very simple request of starting and stopping the logging process to controlling

Table 1 | **Overall assessment of various scientific machine learning benchmarking approaches**

Benchmark	Focus			Process			Challenges			
	Scientific	Application	System	Metrics	Framework	Reporting	Data	Distribution	Coverage	Extensibility
Deep500	None	None	Partial	Full	Full	Partial	None	None	None	Partial
RLBench	None	Partial	Partial	Full	None	Partial	Partial	Partial	Partial	Partial
CORAL-2 (DLS/BDAS)	Partial	Full	Full	Full	Partial	Partial	None	None	Full	None
AIBench + HPC AI500	Full	Full	Full	Full	None	Full	Partial	Partial	Partial	Partial
DAWNBench	None	Full	Full	Full	None	Partial	None	None	None	None
MLCommons Science	Full	Full	Partial	Full	None	Partial	Partial	Partial	Full	Partial
SciMLBench	Full	Full	Full	Full	Full	Partial	Full	Full	Full	Full
Community competitions	Partial	None	None	Partial	None	Partial	Partial	None	Partial	None

In qualitatively assessing how far each approach addresses the concerns, we have indicated whether they offer no support (none), partial or questionable support (partial) or fully support the concern (full).

what is specifically being logged, such as science-specific outputs or domain-specific metrics. Since the logging process includes all relevant details (including the runtime or the power and energy usage, where permitted), the benchmark designer or developer is responsible for deciding on the appropriate metric, depending on the context. For example, it is possible for the developer to rely on a purely scientific metric or to specify a metric to quantify the energy efficiency of the benchmark.

Benchmarking process. With the framework handling most of the complexity of collecting performance data, there is the opportunity to cover a wide range of metrics (even retrospectively, after the benchmarks have been run) and have the ability to control the reporting and compliance through controlled runs. However, it is worth noting that, although the framework can support and collect a wide range of runtime and science performance aspects, the choice is left to the user to decide the ultimate metrics to be reported. For example, the performance data collected by the framework can be used to generate a final figure of merit to compare different ML models or hardware systems for the same problem. The benchmarks can be executed purely using the framework or using containerized environments, such as Docker or Singularity. Although running benchmarks natively using the framework is possible, native code execution on production systems is often challenging and ends up demanding various dependencies. For these reasons, executing these benchmarks on containerized environments is recommended on production, multinode clusters. We have found that the resulting container execution overheads are minimal.

Data curation and distribution. SciMLBench uses a carefully designed curation and distribution mechanism (a process illustrated in FIG. 2), given below.

- Each benchmark has one or more associated datasets. These benchmark–dataset associations are specified through a configuration tool that is not only framework friendly but also interpretable by scientists.
- As the scientific datasets are usually large, they are not maintained along with the code. Instead, they are maintained in a separate object storage, whose exact locations are visible to the benchmarking framework and to users.
- Users downloading benchmarks will only download the reference implementations (code) and not the data. This enables fast downloading of the benchmarks and the framework. Since not all datasets will be of interest to everyone, this approach prevents unnecessary downloading of large datasets.
- The framework takes the responsibility for downloading datasets on demand or when the user launches the benchmarking process.

In addition to these basic operational aspects, the benchmark datasets are stored in an object storage to enable better resiliency and repair mechanisms compared with simple file storage. The datasets are also mirrored in several locations to enable the framework to choose the data source closest to the location of the user. The datasets are also regularly backed up, as they constitute valuable digital assets.

Extensibility and coverage. The overall design of SciMLBench supports several user scenarios: the ability to add new benchmarks with little knowledge of the framework, ease of use, platform interoperability and ease of customization. The design relies on two API calls, which are illustrated in the documentation with a number of toy examples, as well as some practical examples.

Conclusion

In this Perspective, we have highlighted the need for scientific ML benchmarks and explained how they differ from conventional benchmarking initiatives. We have outlined the challenges in developing a suite of useful scientific ML benchmarks. These challenges span a number of issues, ranging from the intended focus of the benchmarks and the benchmarking processes, to challenges around actually developing a useful ML benchmark suite. A useful scientific ML suite must, therefore, go beyond just providing a disparate collection of ML-based scientific applications. The critical aspect here is to provide support for end users not only to be able to effectively use the ML benchmarks but also to enable them to develop new benchmarks and extend the suite for their own purposes.

We overviewed a number of contemporary efforts for developing ML benchmarks, of which only a subset has a focus of ML for scientific applications. Almost none of these initiatives considers the problem of the efficient distribution of large datasets. The majority of the approaches rely on externally sourced datasets, with the implicit assumption that users will take care of the data issues. We discussed in more detail the SciMLBench initiative, which includes a benchmark framework that not only addresses the majority of these concerns but is also designed for easy extensibility.

The characteristics of these ML benchmark initiatives are summarized in TABLE 1, which shows that the benchmarking community has several issues to address to ensure that the scientific community is equipped with the right set of tools to become more efficient in leveraging the use of ML technologies in science.

Code availability statement

The relevant code for the benchmark suite can be found at https://github.com/stfc-sciml/sciml-bench.

Jeyan Thiyagalingam[1], *Mallikarjun Shankar*[2], *Geoffrey Fox*[3] *and Tony Hey*[1] ✉

[1]*Rutherford Appleton Laboratory, Science and Technology Facilities Council, Harwell Campus, Didcot, UK.*

[2]*Oak Ridge National Laboratory, Oak Ridge, TN, USA.*

[3]*Computer Science and Biocomplexity Institute, University of Virginia, Charlottesville, VA, USA.*

✉*e-mail: Tony.Hey@stfc.ac.uk*

https://doi.org/10.1038/s42254-022-00441-7

Published online 6 April 2022

1. Sejnowski, T. J. *The Deep Learning Revolution* (MIT Press, 2018).
2. Hey, T., Butler, K., Jackson, S. & Thiyagalingam, J. Machine learning and big scientific data. *Philos. Trans. R. Soc. A Math. Phys. Eng. Sci.* **378**, 20190054 (2020).
3. Callaway, E. 'It will change everything': DeepMind's AI makes gigantic leap in solving protein structures. *Nature* **588**, 203–204 (2020).
4. Jumper, J. et al. Highly accurate protein structure prediction with AlphaFold. *Nature* **596**, 583–589 (2021).
5. Raissi, M., Perdikaris, P. & Karniadakis, G. E. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *J. Comput. Phys.* **378**, 686–707 (2019).
6. Greydanus, S., Dzamba, M. & Yosinski, J. in *Advances in Neural Information Processing Systems* Vol. 32 (eds. Wallach, H. et al.) (Curran Associates, Inc., 2019).
7. Butler, K., Le, M., Thiyagalingam, J. & Perring, T. Interpretable, calibrated neural networks for analysis and understanding of inelastic neutron scattering data. *J. Phys. Condens. Matter* **33**, 194006 (2021).
8. Hartigan, J. A. & Wong, M. A. A k-means clustering algorithm. *J. R. Stat. Soc. C Appl. Stat.* **28**, 100–108 (1979).
9. Cortes, C. & Vapnik, V. Support-vector networks. *Mach. Learn.* **20**, 273–297 (1995).
10. Baldi, P. in *Proceedings of ICML Workshop on Unsupervised and Transfer Learning* Vol. 27 (eds Guyon, I., Dror, G., Lemaire, V., Taylor, G. & Silver, D.) 37–49 (PMLR, 2012).
11. Sutton, R. S. & Barto, A. G. *Reinforcement Learning: An Introduction* (MIT Press, 2018).
12. Dongarra, J. & Luszczek, P. in *Encyclopedia of Parallel Computing* (ed. Padua, D.) 844–850 (Springer, 2011).
13. Sakalis, C., Leonardsson, C., Kaxiras, S. & Ros, A. in *2016 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)* 101–111 (IEEE, 2016).
14. Bailey, D. H. in *Encyclopedia of Parallel Computing* (ed. Padua, D.) 1254–1259 (Springer, 2011).
15. Petitet, A., Whaley, R., Dongarra, J. & Cleary, A. *HPL–a Portable Implementation of the High-Performance Linpack Benchmark for Distributed-Memory Computers* (ICL-UTK Computer Science Department, 2008).
16. Dongarra, J. & Luszczek, P. in *Encyclopedia of Parallel Computing* (ed. Padua, D.) 2055–2057 (Springer, 2011).
17. Henghes, B., Pettitt, C., Thiyagalingam, J., Hey, T. & Lahav, O. Benchmarking and scalability of machine-learning methods for photometric redshift estimation. *Mon. Not. R. Astron. Soc.* **505**, 4847–4856 (2021).
18. Müller, A., Karathanasopoulos, N., Roth, C. C. & Mohr, D. Machine learning classifiers for surface crack detection in fracture experiments. *Int. J. Mech. Sci.* **209**, 106698 (2021).
19. Ede, J. M. & Beanland, R. Improving electron micrograph signal-to-noise with an atrous convolutional encoder-decoder. *Ultramicroscopy* **202**, 18–25 (2019).
20. Deng, J. et al. in *2009 IEEE Conference on Computer Vision and Pattern Recognition* 248–255 (IEEE, 2009).
21. Krizhevsky, A., Sutskever, I. & Hinton, G. E. ImageNet classification with deep convolutional neural networks. *Commun. ACM* **60**, 84–90 (2017).
22. HPL-AI benchmark. https://hpl-ai.org/.
23. Müller, M., Whitney, B., Henschel, R. & Kumaran, K. in *Encyclopedia of Parallel Computing* (ed. Padua, D.) 1886–1893 (Springer, 2011).
24. Ben-Nun, T. et al. in *2019 IEEE International Parallel and Distributed Processing Symposium (IPDPS)* 66–77 (IEEE, 2019).
25. James, S., Ma, Z., Rovick Arrojo, D. & Davison, A. J. RLBench: The robot learning benchmark & learning environment. *IEEE Robot. Autom. Lett.* **5**, 3019–3026 (2020).
26. CORAL-2 benchmarks. https://asc.llnl.gov/coral-2-benchmarks.
27. Coleman, C. A. et al. in *31st Conference on Neural Information Processing Systems (NIPS 2017)* (2017).
28. BenchCouncil AIBench. https://www.benchcouncil.org/aibench/index.html.
29. MLCommons HPC Benchmark. https://mlcommons.org/en/groups/training-hpc/.
30. Thiyagalingam, J. et al. SciMLBench: A benchmarking suite for AI for science. https://github.com/stfc-sciml/sciml-bench (2021).
31. Kaggle Competitions. https://www.kaggle.com/.
32. Wu, X. et al. in *Proceedings of the 48th International Conference on Parallel Processing* 78 (Association for Computing Machinery, 2019).
33. Jiang, Z. et al. in *2021 IEEE International Conference on Cluster Computing (CLUSTER)* 47–58 (IEEE, 2021).
34. Krizhevsky, A., Nair, V. & Hinton, G. The CIFAR-10 dataset. *Canadian Institute for Advanced Research* http://www.cs.toronto.edu/~kriz/cifar.html (2010).
35. Rajpurkar, P., Zhang, J., Lopyrev, K. & Liang, P. in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing* 2383–2392 (Association for Computational Linguistics, 2016).
36. MLCommons Science. https://mlcommons.org/en/groups/research-science/.
37. Rasp, S. et al. WeatherBench: a benchmark data set for data-driven weather forecasting. *J. Adv. Model. Earth Syst.* **12**, e2020MS002203 (2020).
38. The MAELSTROM Project. https://www.maelstrom-eurohpc.eu/.
39. Cai, L. et al. Surrogate models based on machine learning methods for parameter estimation of left ventricular myocardium. *R. Soc. Open Sci.* **8**, 201121 (2021).
40. Wilkinson, M. D. et al. The FAIR Guiding Principles for scientific data management and stewardship. *Sci. Data* **3**, 160018 (2016).

Acknowledgements
We would like to thank Samuel Jackson, Kuangdai Leng, Keith Butler and Juri Papay from the Scientific Machine Learning Group at the Rutherford Appleton Laboratory, Junqi Yin and Aristeidis Tsaris from Oak Ridge National Laboratory and the MLCommons Science Working Group for valuable discussions. This work was supported by Wave 1 of the UKRI Strategic Priorities Fund under the EPSRC grant EP/T001569/1, particularly the 'AI for Science' theme within that grant, by the Alan Turing Institute and by the Benchmarking for AI for Science at Exascale (BASE) project under the EPSRC grant EP/V001310/1. This research also used resources from the Oak Ridge Leadership Computing Facility, which is a DOE Office of Science user facility supported under contract DE-AC05-00OR22725 and from the Science and Technology Facilities Council, particularly that of the Pearl AI resource.

Author contributions
J.T., M.S., G.F. and T.H. conceptualized the idea of scientific benchmarking. J.T. designed the SciMLBench framework, data architecture and conceptualized the overarching set of features. T.H. has overseen the overall developmental efforts, along with J.T., M.S. and G.F. All authors have contributed towards the writing of the manuscript.

Competing interests
The authors declare no competing interests.

Peer review information
Nature Reviews Physics thanks Tal Ben-Nun, Prasanna Balaprakash and the other, anonymous, reviewer for their contribution to the peer review of this work.

Publisher's note
Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.