The Journal of Open Source Software

PLAID: Physics-Learning AI Datamodel

Fabien Casenave [1], **Xavier Roynard** [1], **and Alexandre Devaux–Rivière** [1,2]

1 SafranTech, Safran Tech, Digital Sciences & Technologies, 78114 Magny-Les-Hameaux, France **2** EPITA, 14-16 Rue Voltaire, 94270 Le Kremlin-Bicêtre, France

DOI: 10.xxxxxx/draft

Software

- Review
- Repository
- Archive

Editor: Vangelis Kourlitis

Reviewers:

- @Nelly-Barret
- @karanprime

Submitted: 11 June 2025
Published: unpublished

Summary

PLAID (Physics-Learning AI Datamodel) is a Python library and data format for representing, storing, and sharing physics simulation datasets for machine learning. Unlike domain-specific formats, PLAID accommodates time-dependent, multi-resolution simulations and heterogeneous meshes. The library provides a high-level API to easily load, inspect, and save data. Beyond basic I/O, PLAID includes utilities for machine-learning workflows. It provides converters to build PLAID datasets from generic tabular data, and a "Hugging Face bridge" to push/pull datasets via the Hugging Face hub. In short, PLAID couples a flexible on-disk standard with a software toolkit to manipulate physics data, addressing the needs of ML researchers in fluid dynamics, structural mechanics, and related fields in a generic fashion. Full documentation, examples and tutorials are available at plaid-lib.readthedocs.io.

Statement of Need

Machine learning for physical systems often suffers from inconsistent data representations across different domains and simulators. Existing initiatives typically target narrow problems: e.g., separate formats for CFD or for finite-element data, and dedicated scripts to process each new dataset. This fragmentation hinders reproducibility and reuse of high-fidelity data.

In practice, simulation datasets for machine-learning workflows are often distributed through general-purpose scientific formats such as HDF5 or visualization-oriented formats such as VTK, combined with project-specific conventions. While several recent benchmark initiatives (e.g., The Well (Ohana et al., 2024), PDEBench (Takamoto et al., 2022), PDEArena (Gupta & Brandstetter, 2022)) standardize tasks and evaluation metrics for physics-informed ML, they typically rely on bespoke data organizations rather than a shared datamodel. As a result, interoperability and reuse across datasets and simulators remain limited.

PLAID addresses this gap by providing a generic, unified datamodel that can describe many physics simulation data. It leverages the CGNS standard (Poinot & Rumsey, 2018) to capture complex geometry and time evolution: for example, CGNS supports multi-block topologies and evolving meshes, with a data model that separates abstract topology (element families, etc.) from concrete mesh coordinates. On top of CGNS, PLAID layers a lightweight organizational structure.

By promoting a common standard, PLAID makes physics data interoperable across projects. It has already been used to package and publish multiple datasets covering structural mechanics and computational fluid dynamics. These PLAID-formatted datasets (hosted on Zenodo and Hugging Face) have supported ML benchmarks, democratizing access to simulation data.

The Journal of Open Source Software

Functionality

- **Data Model and Formats:** A PLAID dataset is organized within a root folder (or archive), distinctly separating simulation data from machine learning task definitions, as illustrated in Figure 1. The `dataset/` directory contains numbered sample subfolders (`sample_000...`), each holding one or more `.cgns` files under `meshes/` and a `scalars.csv` file. The `dataset/infos.yaml` file contains human-readable descriptions and metadata. The `problem_definition/` folder provides machine learning context. It includes `problem_infos.yaml` (specifying the ML task inputs/outputs) and `split.csv` (defining train/test splits). This design supports time evolution and multi-block/multi-geometry problems out of the box.

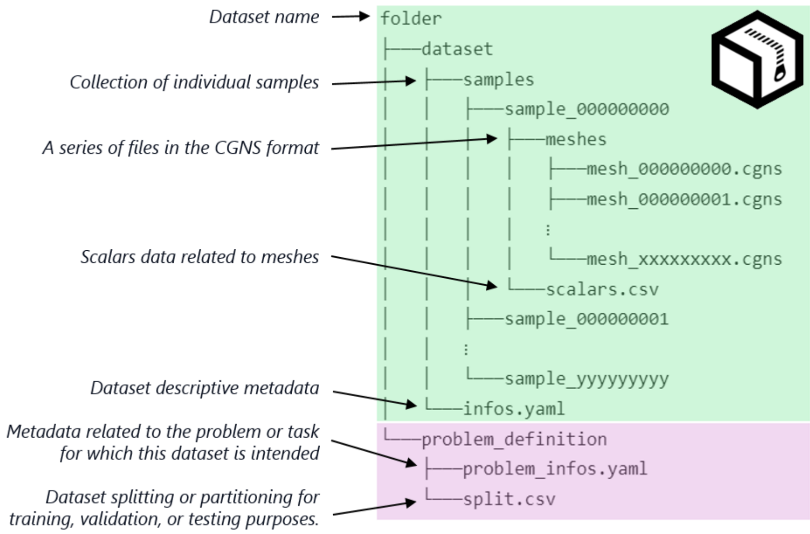

Figure 1: Overview of the PLAID dataset architecture.

- **Supported Data Types:** PLAID handles scalar, time-series and vector field data on meshes, as well as sample-specific metadata. The `get_mesh(time)` method reconstructs the full CGNS tree for a given timestep, with links resolved if requested (thereby returning the complete mesh). Thus PLAID naturally supports mesh-based simulation outputs with arbitrary element types and remeshing between time steps. Heterogeneity is allowed: missing data is supported, and outputs on testing sets may be missing on purpose to facilitate benchmark initiatives.

- **High-Level API:** The top-level `Dataset` class manages multiple `Sample` objects. Users can create an empty `Dataset()` and add samples via `add_sample()`, or load an existing PLAID data archive by calling `Dataset("path_to_plaid_dataset")`. The `Dataset` object summarizes itself (e.g. printing "Dataset(3 samples, 2 scalars, 5 fields)") and provides access to samples by ID. Batch operations are supported: one can `dataset.add_samples(...)` to append many samples, or use the classmethods `Dataset.load_from_dir()` and `load_from_file()` to load data from disk, with optional parallel workers. This high-level interface abstracts away low-level I/O, letting users focus on ML pipelines.

- **Utilities:** PLAID includes helper modules for common tasks in data science workflows. The `plaid.utils.split` module provides a `split_dataset` function to partition data into training/validation/testing subsets according to user-defined ratios. The `plaid.utils.interpolation` module implements piecewise linear interpolation routines to resample time series fields or align datasets with different timesteps. The `plaid.utils.stats` module offers an `OnlineStatistics` class to compute running statistics (min, mean, variance, etc.) on arrays, which can be used to analyze dataset distributions. Moreover, a "Hugging Face bridge" (`plaid.bridges.huggingface_bridge`) enables converting PLAID datasets to/from Hugging Face Dataset objects.

Usage and Applications

PLAID is designed for AI/ML researchers and practitioners working with simulation data. Various datasets, including 2D/3D fluid and structural simulations, are provided in PLAID format in Hugging Face and Zenodo. Interactive benchmarks are hosted in a Hugging Face community on these datasets, providing detailed instructions and PLAID commands for data retrieval and manipulation, see (Casenave et al., 2025). These datasets are also used in recent publications to illustrate the performance of the proposed scientific ML methods. In (Casenave et al., 2024; Kabalan, Casenave, Bordeu, Ehrlacher, & Ern, 2025; Kabalan, Casenave, Bordeu, & Ehrlacher, 2025), Gaussian-process regression methods with mesh morphing are applied to these datasets. In (Carpintero Perez et al., 2024a, 2024b) the datasets are leveraged in graph-kernel regression methods applied to fluid/solid mechanics.

In summary, PLAID provides a comprehensive framework for physics-based ML data. By combining a unified data model, support for advanced mesh features, and helpful utilities, it addresses the need for interoperable, high-fidelity simulation datasets. Future enhancements involve developing general-purpose PyTorch dataloaders compatible with PLAID, along with establishing standardized evaluation metrics and unified pipelines for training and inference using the PLAID framework.

References

Carpintero Perez, R., Da Veiga, S., Garnier, J., & Staber, B. (2024a). Gaussian process regression with Sliced Wasserstein Weisfeiler-Lehman graph kernels. *International Conference on Artificial Intelligence and Statistics*, 1297–1305.

Carpintero Perez, R., Da Veiga, S., Garnier, J., & Staber, B. (2024b). Learning signals defined on graphs with optimal transport and Gaussian process regression. *arXiv Preprint arXiv:2410.15721*. https://doi.org/10.48550/arXiv.2410.15721

Casenave, F., Roynard, X., Staber, B., Akkari, N., Piat, W., Bucci, M. A., Kabalan, A., Nguyen, X. M. V., Saverio, L., Perez, R. C., & others. (2025). Physics-learning AI datamodel (PLAID) datasets: A collection of physics simulations for machine learning. *arXiv Preprint arXiv:2505.02974*. https://doi.org/10.48550/arXiv.2505.02974

Casenave, F., Staber, B., & Roynard, X. (2024). MMGP: A Mesh Morphing Gaussian Process-based machine learning method for regression of physical problems under nonparametrized geometrical variability. *Advances in Neural Information Processing Systems*, *36*.

Gupta, J. K., & Brandstetter, J. (2022). Towards multi-spatiotemporal-scale generalized pde modeling. *arXiv Preprint arXiv:2209.15616*.

Kabalan, A., Casenave, F., Bordeu, F., & Ehrlacher, V. (2025). O-MMGP: Optimal Mesh Morphing Gaussian Process regression for solving PDEs with non-parametric geometric variations. *arXiv Preprint arXiv:2502.11632*. https://doi.org/10.48550/arXiv.2502.11632

Kabalan, A., Casenave, F., Bordeu, F., Ehrlacher, V., & Ern, A. (2025). Elasticity-based morphing technique and application to reduced-order modeling. *Applied Mathematical Modelling*, *141*, 115929. https://doi.org/10.1016/j.apm.2025.115929

Ohana, R., McCabe, M., Meyer, L., Morel, R., Agocs, F., Beneitez, M., Berger, M., Burkhart, B., Dalziel, S., Fielding, D., & others. (2024). The well: A large-scale collection of diverse physics simulations for machine learning. *Advances in Neural Information Processing Systems*, *37*, 44989–45037.

Poinot, M., & Rumsey, C. L. (2018). Seven keys for practical understanding and use of CGNS. *2018 AIAA Aerospace Sciences Meeting*, 1503. https://doi.org/10.2514/6.2018-1503

Takamoto, M., Praditia, T., Leiteritz, R., MacKinlay, D., Alesiani, F., Pflüger, D., & Niepert, M. (2022). Pdebench: An extensive benchmark for scientific machine learning. *Advances in Neural Information Processing Systems*, *35*, 1596–1611.