Physics-Learning AI Datamodel (PLAID) datasets: a collection of physics simulations for machine learning

Anonymous Author(s)
Affiliation
Address
email

Abstract

Machine learning-based surrogate models have emerged as a powerful tool to accelerate simulation-driven scientific workflows. However, their widespread adoption is hindered by the lack of large-scale, diverse, and standardized datasets tailored to physics-based simulations. While existing initiatives provide valuable contributions, many are limited in scope-focusing on specific physics domains, relying on fragmented tooling, or adhering to overly simplistic datamodels that restrict generalization. To address these limitations, we introduce PLAID (Physics-Learning AI Datamodel), a flexible and extensible framework for representing and sharing datasets of physics simulations. PLAID defines a unified standard for describing simulation data and is accompanied by a library for creating, reading, and manipulating complex datasets across a wide range of physical use cases (gitlab.com/drti/plaid). We release six carefully crafted datasets under the PLAID standard, covering structural mechanics and computational fluid dynamics, and provide baseline benchmarks using representative learning methods. Benchmarking tools are made available on Hugging Face, enabling direct participation by the community and contribution to ongoing evaluation efforts (huggingface.co/PLAIDcompetitions).

1 Introduction

Numerical simulation is a cornerstone of scientific and engineering research, providing essential insights into complex physical phenomena across a wide range of domains—including earth and environmental sciences [1], life sciences and medicine [2], finance and economics [3], and industrial engineering [4, 5, 6]. These simulations rely on solving partial differential equations (PDEs) using space and time discretization and numerical methods, typically implemented in large-scale computational solvers. While accurate, these simulations are often computationally intensive, with a single high-fidelity run potentially requiring several hours or days. Many practical scenarios demand solving the same physical model across a wide range of settings—such as in design exploration, optimization, real-time simulation, and uncertainty quantification. In such many-query contexts, reliance on costly simulations becomes impractical. To address this, a broad spectrum of surrogate modeling techniques has been proposed to approximate simulation outputs at a fraction of the computational cost.

Classical surrogate models perform non-linear regression over parametric spaces using statistical learning techniques, such as polynomial regression, nearest neighbors, support vector machines, random forests [7], and Gaussian processes [8]. These models are widely supported by software libraries such as UQLab [9], OpenTURNS [10], Dakota [11] and Lagun [12]. However, they are typically restricted to low-dimensional, tabular parameter spaces and cannot be directly used in more complex simulation setups. In contrast, many modern applications involve richer input configurations, including unstructured geometries, spatially varying fields, and complex boundary or material conditions. These settings require learning from heterogeneous, high-dimensional data with nonparametric variability.

Recent advances in scientific machine learning have begun to address these challenges. One line of work, often referred to as physics-based model reduction, builds surrogates that approximate the solution of the governing equations directly [13, 14, 15, 16, 17, 18]. Other approaches have also been proposed using non-parametric methods based on the use of morphing [19, 20, 21] or optimal transport [22, 23], and have the advantage of requiring a smaller number of design points. Increasingly, deep learning methods—particularly Graph Neural Networks (GNNs)—have shown promise in capturing the spatiotemporal dynamics of physical systems. Building on the message-passing paradigm introduced in [24], architectures such as MeshGraphNets [25] extend GNNs to general mesh-based simulations. Hierarchical versions like MultiScale MeshGraphNets [26] enhance scalability and accuracy, while recent works demonstrate effectiveness in inverse [27] and steady-state problems [28]. Other developments include geodesic convolutions [29], multi-resolution models [30, 31], and improved pooling strategies [32]. Tools such as PhysicsNeMo [33], PyTorch Geometric [34], and Deep Graph Library [35] provide convenient foundations for implementing these methods.

Despite these advances, widespread adoption remains hindered by a critical bottleneck: the lack of large-scale, diverse, and standardized datasets for training and benchmarking. Existing datasets often cover narrow physical regimes, rely on ad hoc formats, or are tied to specific libraries—limiting reusability and interoperability. Furthermore, many datasets are tailored to isolated challenges (e.g., time dependence) but fail to accommodate others (e.g., geometric variation). This fragmentation is particularly problematic in the context of recent developments in physics foundation models [36, 37, 38, 39, 40], which require large, flexible, and standardized sources of training data.

To address these limitations, we introduce PLAID (Physics-Learning AI Datamodel), a comprehensive framework for representing and manipulating datasets of physics simulations for machine learning. PLAID defines a generic, extensible datamodel that supports a wide range of use cases—including time-dependent problems, remeshing, mixed-element unstructured meshes, node/element tagging, multiple spatial dimensions and topologies. We provide an accompanying software library to facilitate dataset creation, reading, and high-level interaction, that can leverage Hugging Face infrastructure for efficient streaming, caching, and sharing.

In Section 2, we review relevant dataset efforts in the literature. Section 3 introduces the PLAID datamodel and implementation, along with six publicly released datasets in structural mechanics and computational fluid dynamics, presented in Section 4, that showcase rich variability in physics and numerical complexity. In Section 5, we provide performance benchmarks across a range of machine learning methods, hosted on Hugging Face to allow community participation and continual updates. We conclude with perspectives in Section 6.

2 Related Work

Progress in machine learning has been largely driven by the availability of large, diverse, and carefully curated datasets [41, 42, 43]. Natural language processing models are trained on web-scale data [44, 45, 46, 47], and vision models routinely leverage billions of image–text pairs [48, 49, 50].

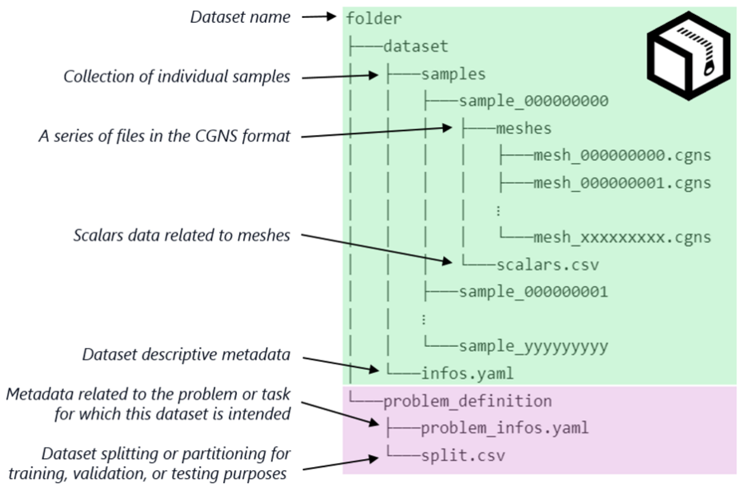
In contrast, datasets for physics learning remain comparatively underdeveloped. Early benchmarks targeted standard physics problems and reference simulations [51, 52, 53]. More recent datasets have focused on complex, domain-specific settings [54, 55, 56, 57, 58, 59, 60, 61, 62]. The recently proposed Well [63] includes an impressive list of datasets for various physics, but is limited to structured grids (uniformly sampled domains).

Structural mechanics simulations, with non-linear constitutive laws, are of paramount importance for industrial design, and are poorly represented in available datasets. Most available datasets use a datamodel that limit their evolution and generality. Complex industrial settings include vertices and element tags, heterogeneous data with remeshing, multiple meshes of various dimensions, topologies and mixed element types, compatible with commercial codes routinely used by design engineers. Besides, most datasets come with a library dedicated to the dataset, featuring specific commands and hypothesis, which limit they wide adoption.

3 PLAID standard

We propose PLAID, a datamodel for datasets for machine learning applied to physics-related problems. PLAID also refers to the library that implements this datamodel, available on GitLab [64], and to the file format used to store data. The primary goal of the library is to provide a general and flexible framework for defining physics-based dataset, along with a corresponding learning task. The datamodel is built on CGNS [65], leveraging its well-established formalization of a wide range of physical configurations.

PLAID datasets are provided either as human-readable data storage, or stored using Hugging Face datasets tools [66]. In the former case, YAML and CSV files can be opened with any text editor, while CGNS files containing physical configurations can be visualized using tools such as ParaView, see Figure 1. In the latter, we benefit directly from powerful data management such as caching and online streaming.


Dataset name
Collection of individual samples
A series of files in the CGNS format
Scalars data related to meshes
Dataset descriptive metadata
Metadata related to the problem or task for which this dataset is intended
Dataset splitting or partitioning for training, validation, or testing purposes

```
folder
├──dataset
│   ├──samples
│   │   ├──sample_000000000
│   │   │   ├──meshes
│   │   │   │   ├──mesh_000000000.cgns
│   │   │   │   ├──mesh_000000001.cgns
│   │   │   │   ⋮
│   │   │   │   └──mesh_xxxxxxxxx.cgns
│   │   │   └──scalars.csv
│   │   ├──sample_000000001
│   │   ⋮
│   │   └──sample_yyyyyyyyy
│   └──infos.yaml
└──problem_definition
    ├──problem_infos.yaml
    └──split.csv
```

Figure 1: PLAID files structure.

Additionally, PLAID offers high-level utilities for constructing, handling and read/write datasets efficiently. Documentation is available online, with usage examples and tutorials showing how one can create a PLAID dataset from its own data. We also mention Muscat, a finite element toolbox available on GitLab [67, 68], containing various reader and writers from and to various files formats used in numerical simulation codes for physics, and routines to generate the CGNS data structures used in PLAID. Samples can feature multiple meshes, scalars, fields and time series. We illustrate how PLAID can deal with complex heterogeneous data by explaining some available commands:

- `dataset[0].get_field_names(name, zone_name, base_name, location, time)`: retrieves the first sample's field called `name`, for chosen `zone_name`, `base_name`, `location` (Vertex, CellCenter, FaceCenter, ...) and `time` in the CGNS structure. Fields and meshes can change over time, allowing remeshing and field appearance/disappearance at any time step.

- `sample.get_field(name)`: automatic handling of default values to prevent exposing `zone_name`, `base_name` and `location` arguments to simple cases with no ambiguity.

- `sample.get_mesh(apply_links = True)`: allows to link meshes between CGNS data structures to prevent multiple copies in case of fixed mesh cases.

More examples are provided in Appendix B.

4 PLAID datasets

4.1 Structural mechanics

4.1.1 `Tensile2d` [69] (Zenodo, Hugging Face)

`Tensile2d` is a simple dataset of 2D quasistatic non-linear structural mechanics simulations, in small deformations and plane strain regimes, solved with Z-set [70] using the finite element method. The material is modeled with a non-linear constitutive law. The dataset computes the deformation of a structure subjected to an imposed negative constant pressure at the top, and zero displacement at the bottom, see Figure 2 (left). Only the steady-state solution is kept.

Input variability in the dataset are the unstructured meshes (variable shape, number of nodes and connectivity), the pressure P at the top boundary condition (scalar) and 5 scalars modeling the non-linear constitutive law: (`p1`, `p2`, `p3`, `p4` and `p5`). Outputs of interest are 4 scalars (`max_von_mises`, `max_q`, `max_U2_top` and `max_sig22_top`) and 6 fields (`U1`, `U2`, `q`, `sig11`, `sig22` and `sig12`). Seven nested training sets are provided, as well as a testing set and two out-of-distribution samples.

4.1.2 `2D_MultiScHypEl` [71] (Zenodo, Hugging Face)

`2D_MultiScHypEl`, standing for 2D multiscale hyperelasticity, is a dataset of 2D quasistatic non-linear structural mechanics simulations under large deformation and plane strain conditions, solved with DOLFINx [72] using the finite element method. The material behavior follows a compressible hyperelastic constitutive law, capturing complex non-linear responses. Each simulation corresponds to the homogenization of a porous representative volume element (RVE), subject to kinematically uniform boundary conditions (KUBC) [73], see Figure 2 (right).

Input variability in the dataset are the unstructured meshes (variable shape, number of nodes, connectivity and topology–the number of circular inclusions) and the 3 scalars modeling the KUBC, namely the components `C11`, `C12`, and `C22` of the macroscopic right Cauchy-Green deformation tensor. Outputs of interest are 1 scalar (`effective_energy`) and 7 fields (displacements `u1`, `u2`; first Piola-Kirchhoff stress components `P11`, `P12`, `P22`, `P21` and the strain energy density field `psi`). Various training and testing sets are provided (both across all topologies and within each topology class).

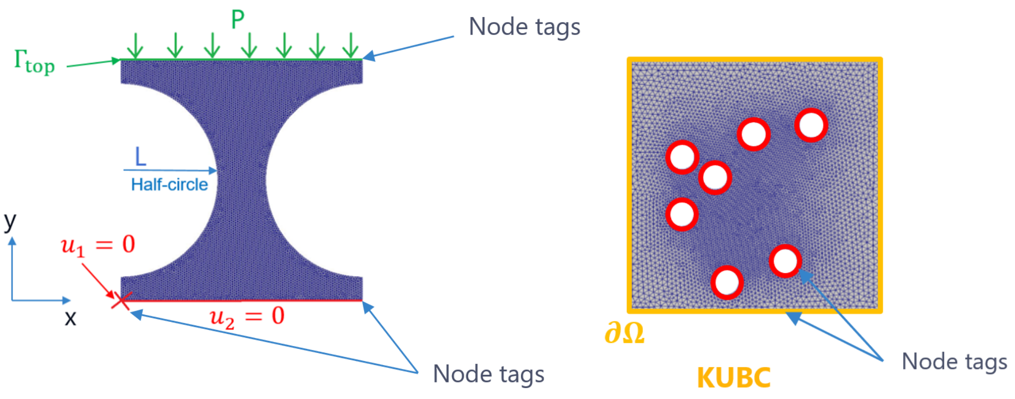


Figure 2: Settings for `Tensile2d` (left) and `2D_MultiScHypEl` (right).

4.1.3 `2D_ElPlDynamics` [74] (Zenodo, Hugging Face)

`2D_ElPlDynamics`, standing for 2D elasto-plasto dynamics, is a dataset of 2D dynamic non-linear structural mechanics simulations, in large deformations and plane strain regimes, solved with Open-Radioss [75] using the finite element method. The material is modeled with a non-linear elastoplastic law, with damage (modeled using element erosion), failure and a non-local method for reducing mesh sensitivity. The dataset computes the transient deformation of a 2D structure, subjected to imposed displacement on the right and zero displacement on the left, see Figure 3 (left).

Input variability in the dataset are the unstructured meshes (variable shape, number of nodes, connectivity and topology). Outputs of interest are 3 fields (`U_x` and `U_y` the displacement fields at the nodes, and EROSION_STATUS a boolean field at the elements describing the state – valid or broken – of each element). A training and a testing set are provided.

4.2 Computational fluid mechanics

4.2.1 Rotor37 [76] (Zenodo, Hugging Face)

Rotor37 is a dataset of 3D compressible steady-state Reynolds-Averaged Navier-Stokes (RANS) simulations, solved with elsA [77] using the finite volumes method. Large scale simulations around the rotor37 blade inside a 3D duct have been computed, with inflow, outflow and periodic boundary conditions. An adequate turbulence model and laws of the wall have been chosen. The dataset only keeps the steady-state solution at the boundary of the blade inside the duct, and scalars of interest, see Figure 3 (right).

Input variability in the dataset are the block-structured anisotropic meshes (variable shape, normals at the blade surface are provided) and 2 scalars (the pressure P and the rotation speed Omega of the blade). Outputs of interest are 3 scalars (Massflow, Compression_ratio and Efficiency) and 3 fields (Density, Pressure, Temperature). Eight nested training sets and a testing set are provided.

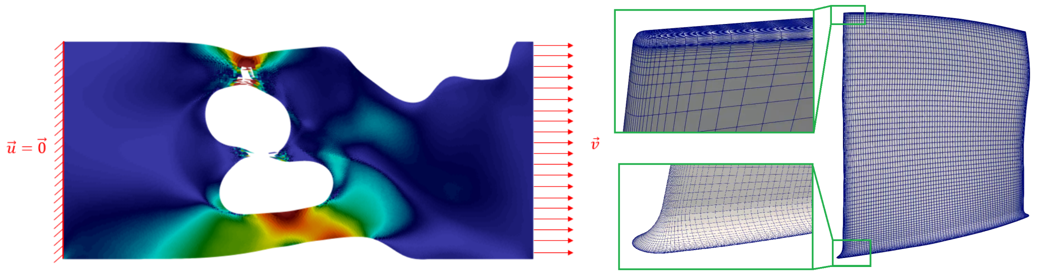


Figure 3: Settings for 2D_ElPlDynamics (left) and Rotor37 (right).

4.2.2 2D_profile [78] (Zenodo, Hugging Face)

2D_profile is a dataset of 2D compressible steady-state Reynolds-Averaged Navier-Stokes (RANS) simulations, solved with elsA [77] using the finite volumes method. The flow is computed around 2D profiles, which present large deformation around shapes resembling airfoils or propeller blades, on a large refined meshes, with inflow, outflow and periodic boundary conditions, at a supersonic regime. An adequate turbulence model and laws of the wall have been chosen. The dataset only keeps the steady-state solution on a zone cropped close to the profile, see Figure 4 (left).

Input variability in the dataset are the unstructured anisotropic meshes (variable shape, number of nodes and connectivity). Outputs of interest are 4 fields (Mach, Pressure, Velocity-x and Velocity-y). A training and a testing set are provided.

4.2.3 VKI-LS59 [79] (Zenodo, Hugging Face)

VKI-LS59 is a dataset of 2D compressible steady-state Reynolds-Averaged Navier-Stokes (RANS) simulations, solved with BROADCAST [80] using the finite volumes method with high-order corrections. The flow is computed around the VKI-LS59 blade, with inflow, outflow and periodic boundary conditions. A Spalart-Allmaras turbulence model has been chosen, see Figure 4 (right).

Input variability in the dataset are the block-structured anisotropic meshes (variable shape, number of nodes and connectivity, the distance field to the blade surface is provided) and 2 scalars (angle_in and mach_out). Outputs of interest are 6 scalars (Q, power, Pr, Tr, eth_is and angle_out) and 7 fields (ro, rou, rov, roe, nut, mach and M_iso – this last being only defined at the surface of the blade). Eight nested training sets are provided, as well as a testing set.

4.2.4 AirfRANS [55]

AirfRANS is a dataset of external aerodynamics, featuring steady-state Reynolds-Averaged Navier-Stokes (RANS) simulations over airfoils at a subsonic regime, proposed in [55], which we refer to for

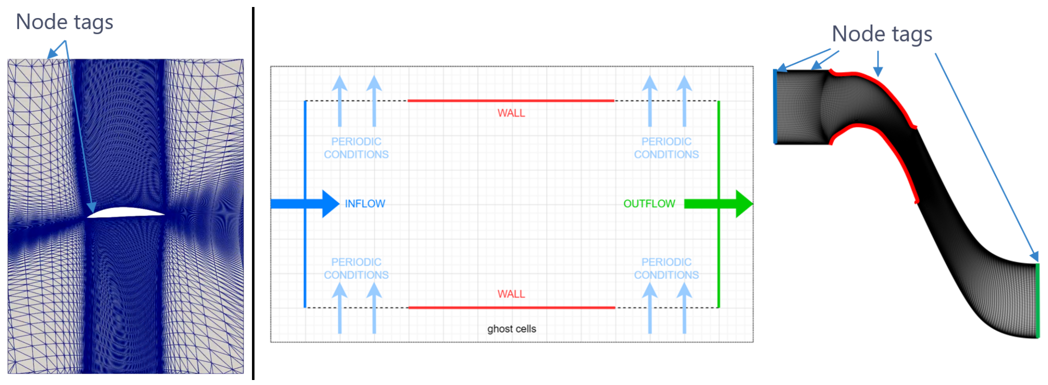


Figure 4: Settings for `2D_profile` (left) and `VKI-LS59` (right).

a detailed description. In addition to the six original datasets, we provide three variants of `AirfRANS` in PLAID format: original [81](Zenodo, Hugging Face), clipped [82](Zenodo, Hugging Face) and remeshed [83](Zenodo, Hugging Face).

Input variability in the dataset are the anisotropic meshes (variable shape, number of nodes and connectivity, the distance field to the airfoil surface is provided) and 2 scalars (`angle_of_attack` and `inlet_velocity`). Outputs of interest are 2 scalars (`C_D` and `C_L`) and 4 fields (`nut`, `Ux`, `Uy` and `p`). Various training and testing sets are provided.

4.3 Dataset collection

The collection of proposed datasets is available online in a Zenodo community and a Hugging Face community. Description summaries are provided in Tables 1 and 2. The collection will be enriched in the future with additional datasets. Since these datasets have been constructed with the goal to provide open benchmarks to the community, the outputs are not provided on the testing sets – but we provide tools to evaluate scores on these testing sets. Some field outputs are illustrated in Table 3.

`Dataset`	Simulation code	Model	Nb samples	Volume Zenodo	Volume HF
`Tensile2d`	Z-set	2D quasistatic non-linear structural mechanics, small deformations, non-linear constitutive law	702	290 MB	383 MB
`2D_MultiScHypEl`	DOLFINx	2D quasistatic non-linear structural mechanics, finite elasticity	1,140	350 MB	419 MB
`2D_ElPlDynamics`	OpenRadioss	2D dynamic non-linear structural mechanics, non-linear non-local constitutive law	1,018	5.7 GB	8.6 GB
`Rotor37`	elsA	3D Navier-Stokes (RANS)	1,200	3.3 GB	4.0 GB
`2D_profile`	elsA	2D Navier-Stokes (RANS)	400	660 MB	814 MB
`VKI-LS59`	BROADCAST	2D Navier-Stokes (RANS)	839	1.9 GB	2.3 GB
`AirfRANS original`	OpenFOAM	2D Navier-Stokes (RANS)	1,000	9.3 GB	15.6 GB
`AirfRANS clipped`				9.7 GB	18.2 GB
`AirfRANS remeshed`				520 MB	611 MB

Table 1: Dataset collection description: model and simulation volume.

5 Benchmark

We first mention that we do not provide benchmark tools and results for the `AirfRANS` datasets, since outputs are public on the testing sets, and various benchmarks are already available in articles [19, 55] and in a competition at NeurIPS 2024 [84].

Dataset	Mesh (mean nodes)	Inputs	Outputs	Splits (train/test)
Tensile2d	tri (9,428)	mesh, 6 scalars	4 scalars, 6 fields	500 / 200
2D_MultiScHypEl	tri (5,692)	mesh, 3 scalars	1 scalar, 7 fiels	764 / 376
2D_ElPlDynamics	tri (25,429)	mesh	3 fields	1,000 / 18
Rotor37	quad (29,773*)	mesh, 2 scalars	4 scalars, 3 fields	1,000 / 200
2D_profile	tri (37,042)	mesh	4 fields	300 / 100
VKI-LS59	quad (36,421*)	mesh, 2 scalars	6 scalars, 7 fields	671 / 168
AirfRANS original	quad (179,776)			
AirfRANS clipped	tri (179,779)	mesh, 2 fields	2 scalars, 4 fields	various splits
AirfRANS remeshed	tri (7,624)			

Table 2: Dataset collection description: data and splits, a $*$ in the second column means that the number of nodes and connectivity are constant in the dataset – the position of the nodes still varies.

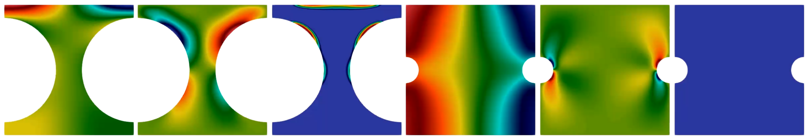
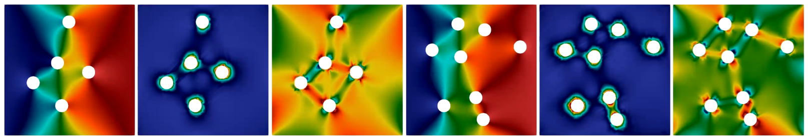
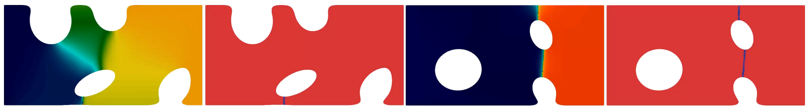
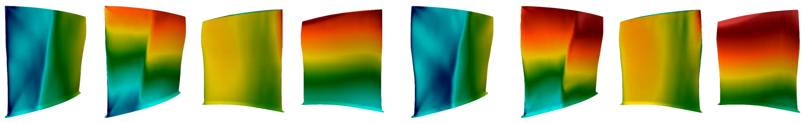
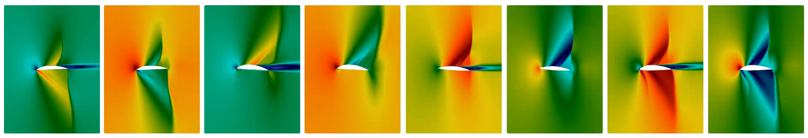
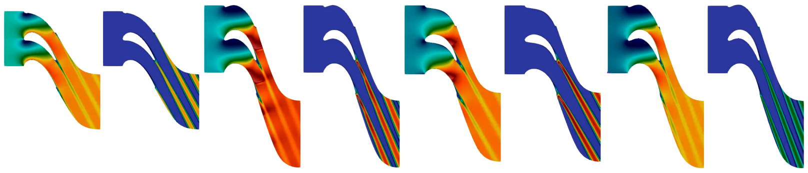
Dataset	Examples of field outputs
Tensile2d	
2D_MultiScHypEl	
2D_ElPlDynamics	
Rotor37	
2D_profile	
VKI-LS59	

Table 3: Dataset collection examples of field outputs illustrations.

5.1 Methods

For practical reasons, we limit the benchmarks to the few following methods:

- MeshGraphNets (MGNs) [25] are GNNs that utilize an encode-process-decode architecture, transforming mesh data into graph structures, processing them through message passing, and decoding the results to predict field outputs.

- Mesh Morphing Gaussian Processes (MMGP) [19] rely on mesh morphing, finite element interpolation and dimensionality reduction to pretreat mesh-based data into a low dimensional embedding, and utilizes Gaussian processes to predict output scalars and fields.
- Vi-Transformer [85] and Augur [1] rely on mesh partitioning to build tokens related to local mesh information and utilize a transformer to predict scalar and field outputs.
- Domain Agnostic Fourier Neural Operators (DAFNO) [86] handle problems involving irregular geometries and evolving domains. It incorporates a smoothed characteristic function into the integral layer architecture of FNOs, allowing the use of Fast Fourier Transform (FFT) for efficient computations while explicitly encoding geometric information.
- Modulated Aerodynamic Resolution Invariant Operator (MARIO) [87] builds upon [88] and exploits implicit neural representations, which model continuous signals by mapping input coordinates directly to output values, without relying on discrete grids or explicit storage.

For more details on the methods and their respective advantages/drawbacks, refer to Appendix A.

5.2 Evaluation metric

Accuracy of the trained models is evaluated by computing RRMSEs (Relative Root Mean Square Errors). Let $\{\mathbf{f}^i_{\text{ref}}\}_{i=1}^{n_\star}$ and $\{\mathbf{f}^i_{\text{pred}}\}_{i=1}^{n_\star}$ be respectively the reference and prediction of a field output on the testing set. The RRMSE is defined as

$$\mathrm{RRMSE}_f(\mathbf{f}_{\text{ref}}, \mathbf{f}_{\text{pred}}) = \left(\frac{1}{n_\star} \sum_{i=1}^{n_\star} \frac{\frac{1}{N^i} \|\mathbf{f}^i_{\text{ref}} - \mathbf{f}^i_{\text{pred}}\|_2^2}{\|\mathbf{f}^i_{\text{ref}}\|_\infty^2} \right)^{1/2},$$

where N^i is the number of nodes in the mesh of sample i, $n_\star$ is the number of samples in the testing set, and $\|\mathbf{f}^i_{\text{ref}}\|_\infty$ is the maximum component in the vector $\mathbf{f}^i_{\text{ref}}$. Similarly for scalar outputs, the following relative RMSE is computed:

$$\mathrm{RRMSE}_s(\mathbf{s}_{\text{ref}}, \mathbf{s}_{\text{pred}}) = \left(\frac{1}{n_\star} \sum_{i=1}^{n_\star} \frac{|s^i_{\text{ref}} - s^i_{\text{pred}}|^2}{|s^i_{\text{ref}}|^2} \right)^{1/2}.$$

The score of a submission, `total_error`, is the mean over fields and scalars RRMSEs.

5.3 Benchmark results

All individual RRMSE and `total_error` for each method applied to each dataset are reported in Table 4. These results are considered neither exhaustive, nor definitive.

We provide the community with online benchmarking applications hosted on Hugging Face as competitions with no end date, see Hugging Face benchmark collection. Each benchmark comes with a visualization application of the datasets, a description of inputs and outputs and detailed instructions for accessing the data and constructing a prediction file. Anyone can register and submit a prediction: submissions are automatically ranked based on `total_error` as defined in Section 5.2. Hence, the benchmark results presented here will be updated in the future. See Section C for additional details on the benchmarking applications.

We notice that MMGP, Vi-Transformer/Augur and MARIO models perform particularly well on our steady-state datasets, while DAFNO has only been evaluated on our unique time-dependent dataset.

6 Conclusion and perspectives

Limitations. PLAID is designed to accommodate a wide range of complex scenarios and remains adaptable to emerging use cases that may not be fully addressed by the current datamodel. We plan to expand our collection with more diverse and large-scale datasets of industrial relevance, complemented by benchmarking applications accessible to the community.

Roadmap. Future developments include the creation of generic PyTorch dataloaders for PLAID, and the standardization of evaluation metrics and training/inference pipelines based on PLAID.

[1]commercial solution from Augur company

Field, *scalar* output	MGN	MMGP	Vi-Transf.	Augur	DAFNO	MARIO
`Tensile2d`						
U1	0.0788	**0.0016**	0.0344	0.0093	-	-
U2	0.1237	**0.0013**	0.0424	0.0135	-	-
sig11	0.1726	**0.0037**	0.0715	0.0187	-	-
sig22	0.0560	**0.0015**	0.0341	0.0099	-	-
sig12	0.0570	**0.0026**	0.0494	0.0121	-	-
max_von_mises	0.0185	**0.0050**	0.0145	0.0219	-	-
max_U2_top	0.0292	**0.0042**	0.0210	0.0344	-	-
max_sig22_top	0.0030	**0.0016**	0.0022	0.0030	-	-
total_error	0.0673	**0.0027**	0.0337	0.0154	-	-
`2D_MultiScHypEl`						
u1	0.0400	-	0.0350	**0.0140**	-	-
u2	0.0444	-	0.0356	**0.0164**	-	-
P11	0.0383	-	0.0611	**0.0185**	-	-
P12	0.0670	-	0.1016	**0.0316**	-	-
P22	0.0383	-	0.0614	**0.0189**	-	-
P21	0.0663	-	0.1005	**0.0311**	-	-
psi	0.0443	-	0.0580	**0.0239**	-	-
effective_energy	0.0111	-	**0.0108**	0.0313	-	-
total_error	0.0437	-	0.0580	**0.0232**	-	-
`2D_ElPlDynamics`						
U_x	-	-	-	-	**0.0025**	-
U_y	-	-	-	-	**0.0291**	-
total_error	-	-	-	-	**0.0158**	-
`Rotor37`						
Density	0.0114	**0.0039**	0.0370	0.0055	-	-
Pressure	0.0114	**0.0039**	0.0366	0.0053	-	-
Temperature	0.0024	**0.0009**	0.0074	0.0012	-	-
Massflow	0.0061	**0.0007**	0.0058	0.0028	-	-
Compression_ratio	0.0060	**0.0007**	0.0055	0.0028	-	-
Efficiency	0.0071	**0.0009**	0.0067	0.0019	-	-
total_error	0.0074	**0.0019**	0.0165	0.0033	-	-
`2D_profile`						
Mach	0.0604	**0.0514**	0.0699	-	-	-
Pressure	0.0466	**0.0335**	0.0430	-	-	-
Velocity-x	0.0735	**0.0585**	0.0854	-	-	-
Velocity-y	0.0566	**0.0483**	0.0570	-	-	-
total_error	0.0593	**0.0480**	0.0638	-	-	-
`VKI-LS59`						
nut	0.1656	0.0822	0.1489	0.0641	-	**0.0259**
mach	0.0451	0.0309	0.0643	0.0245	-	**0.0112**
Q	0.0716	**0.0023**	0.0228	0.0076	-	0.0052
power	0.0403	**0.0057**	0.0168	0.0108	-	0.0077
Pr	0.0068	0.0026	0.0042	0.0050	-	**0.0018**
Tr	0.0001	**0.0000**	0.0001	**0.0000**	-	**0.0000**
eth_is	0.1912	0.1224	0.1311	0.1732	-	**0.0453**
angle_out	0.0263	0.0033	0.0061	0.0040	-	**0.0023**
total_error	0.0684	0.0312	0.0493	0.0362	-	**0.0124**

Table 4: RRMSE and total_error on PLAID benchmarks, best on each line is **bold**, second best is underlined.

References

[1] Jerald A Brotzge, Don Berchoff, DaNa L Carlis, Frederick H Carr, Rachel Hogan Carr, Jordan J Gerth, Brian D Gross, Thomas M Hamill, Sue Ellen Haupt, Neil Jacobs, et al. Challenges and opportunities in numerical weather prediction. *Bulletin of the American Meteorological Society*, 104(3):E698–E705, 2023.

[2] Marco Viceconti and Peter Hunter. The virtual physiological human: ten years after. *Annual review of biomedical engineering*, 18(1):103–123, 2016.

[3] Egle Samanidou, Elmar Zschischang, Dietrich Stauffer, and Thomas Lux. Agent-based models of financial markets. *Reports on Progress in Physics*, 70(3):409, 2007.

[4] Olgierd Cecil Zienkiewicz and Robert Leroy Taylor. *The finite element method for solid and structural mechanics*. Elsevier, 2005.

[5] William H Press. *Numerical recipes 3rd edition: The art of scientific computing*. Cambridge university press, 2007.

[6] Olgierd Cecil Zienkiewicz and PB Morice. *The finite element method in engineering science*, volume 1977. McGraw-hill London, 1971.

[7] Leo Breiman. Random forests. *Machine learning*, 45:5–32, 2001.

[8] C.K.I. Williams and C.E. Rasmussen. *Gaussian processes for machine learning*, volume 2. MIT press Cambridge, MA, 2006.

[9] Stefano Marelli and Bruno Sudret. *UQLab: A Framework for Uncertainty Quantification in Matlab*, pages 2554–2563. 2015.

[10] Michaël Baudin, Anne Dutfoy, Bertrand Iooss, and Anne-Laure Popelin. *OpenTURNS: An Industrial Software for Uncertainty Quantification in Simulation*, pages 1–38. Springer International Publishing, Cham, 2016.

[11] B. M. Adams, W. J. Bohnhoff, K. R. Dalbey, M. S. Ebeida, J. P. Eddy, M. S. Eldred, R. W. Hooper, P. D. Hough, K. T. Hu, J. D. Jakeman, M. Khalil, K. A. Maupin, J. A. Monschke, E. E. Prudencio, E. M. Ridgway, P. Robbe, A. A. Rushdi, D. T. Seidl, J. A. Stephens, L. P. Swiler, and J. G. Winokur. Dakota 6.21.0 documentation. Technical Report SAND2024-15492O, Sandia National Laboratories, Albuquerque, NM, November 2024. Available online from `http://snl-dakota.github.io`.

[12] Sébastien Da Veiga, Clément Bénard, Thierry Gonon, et al. Lagun. `https://gitlab.com/drti/lagun`, 2025.

[13] David Amsallem, Matthew Zahr, Youngsoo Choi, and Charbel Farhat. Design optimization using hyper-reduced-order models. *Structural and Multidisciplinary Optimization*, 51(4):919–940, 2015.

[14] F. Casenave, N. Akkari, F. Bordeu, C. Rey, and D. Ryckelynck. A nonintrusive distributed reduced-order modeling framework for nonlinear structural mechanics—application to elastoviscoplastic computations. *International Journal for Numerical Methods in Engineering*, 121(1):32–53, 2020.

[15] T. Daniel, F. Casenave, N. Akkari, and D. Ryckelynck. Model order reduction assisted by deep neural networks (rom-net). *Advanced Modeling and Simulation in Engineering Sciences*, 7:1–27, 2020.

[16] Kookjin Lee and Kevin T Carlberg. Model reduction of dynamical systems on nonlinear manifolds using deep convolutional autoencoders. *Journal of Computational Physics*, 404:108973, 2020.

[17] Youngkyu Kim, Youngsoo Choi, David Widemann, and Tarek Zohdi. A fast and accurate physics-informed neural network reduced order model with shallow masked autoencoder. *Journal of Computational Physics*, 451:110841, 2022.

[18] Nicolas Barral, Tommaso Taddei, and Ishak Tifouti. Registration-based model reduction of parameterized PDEs with spatio-parameter adaptivity. *Journal of Computational Physics*, 499:112727, 2024.

[19] Fabien Casenave, Brian Staber, and Xavier Roynard. MMGP: a Mesh Morphing Gaussian Process-based machine learning method for regression of physical problems under non-parametrized geometrical variability. *Advances in Neural Information Processing Systems*, 36, 2024.

[20] Abbas Kabalan, Fabien Casenave, Felipe Bordeu, Virginie Ehrlacher, and Alexandre Ern. Elasticity-based morphing technique and application to reduced-order modeling. *Applied Mathematical Modelling*, 141:115929, 2025.

[21] Abbas Kabalan, Fabien Casenave, Felipe Bordeu, and Virginie Ehrlacher. O-MMGP: Optimal Mesh Morphing Gaussian Process regression for solving PDEs with non-parametric geometric variations. *arXiv preprint arXiv:2502.11632*, 2025.

[22] Raphaël Carpintero Perez, Sébastien da Veiga, Josselin Garnier, and Brian Staber. Learning signals defined on graphs with optimal transport and Gaussian process regression, 2025.

[23] Raphaël Carpintero Perez, Sébastien Da Veiga, Josselin Garnier, and Brian Staber. Gaussian process regression with Sliced Wasserstein Weisfeiler-Lehman graph kernels. In *International Conference on Artificial Intelligence and Statistics*, pages 1297–1305. PMLR, 2024.

[24] J. Gilmer, S.S. Schoenholz, P.F. Riley, O. Vinyals, and G.E. Dahl. Neural message passing for quantum chemistry. In *International conference on machine learning*, pages 1263–1272. PMLR, 2017.

[25] T. Pfaff, M. Fortunato, A. Sanchez-Gonzalez, and P.W. Battaglia. Learning mesh-based simulation with graph networks. In *International Conference on Learning Representations*, 2021.

[26] M. Fortunato, T. Pfaff, P. Wirnsberger, A. Pritzel, and P. Battaglia. Multiscale meshgraphnets. *arXiv preprint arXiv:2210.00612*, 2022.

[27] K.R. Allen, T. Lopez-Guevara, K. Stachenfeld, A. Sanchez-Gonzalez, P. Battaglia, J. Hamrick, and T. Pfaff. Physical design using differentiable learned simulators. *arXiv preprint arXiv:2202.00728*, 2022.

[28] L. Harsch and S. Riedelbauch. Direct prediction of steady-state flow fields in meshed domain with graph networks. *arXiv preprint arXiv:2105.02575*, 2021.

[29] P. Baque, E. Remelli, F. Fleuret, and P. Fua. Geodesic convolutional shape optimization. In *International Conference on Machine Learning*, pages 472–481. PMLR, 2018.

[30] M. Lino, C. Cantwell, A.A. Bharath, and S. Fotiadis. Simulating continuum mechanics with multi-scale graph neural networks. *arXiv preprint arXiv:2106.04900*, 2021.

[31] M. Lino, S. Fotiadis, A.A. Bharath, and C.D. Cantwell. Multi-scale rotation-equivariant graph neural networks for unsteady eulerian fluid dynamics. *Physics of Fluids*, 34(8), 2022.

[32] Y. Cao, M. Chai, M. Li, and C. Jiang. Efficient learning of mesh-based physical simulation with bi-stride multi-scale graph neural network. 2023.

[33] PhysicsNeMo Contributors. Nvidia physicsnemo: An open-source framework for physics-based deep learning in science and engineering. `https://github.com/NVIDIA/physicsnemo`, February 2023.

[34] Matthias Fey and Jan E. Lenssen. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

[35] Minjie Wang, Da Zheng, Zihao Ye, Quan Gan, Mufei Li, Xiang Song, Jinjing Zhou, Chao Ma, Lingfan Yu, Yu Gai, Tianjun Xiao, Tong He, George Karypis, Jinyang Li, and Zheng Zhang. Deep graph library: A graph-centric, highly-performant package for graph neural networks. *arXiv preprint arXiv:1909.01315*, 2019.

[36] Jingmin Sun, Yuxuan Liu, Zecheng Zhang, and Hayden Schaeffer. Towards a foundation model for partial differential equations: Multioperator learning and extrapolation. *Phys. Rev. E*, 111:035304, Mar 2025.

[37] Michael McCabe, Bruno Régaldo-Saint Blancard, Liam Parker, Ruben Ohana, Miles Cranmer, Alberto Bietti, Michael Eickenberg, Siavash Golkar, Geraud Krawezik, Francois Lanusse, Mariel Pettee, Tiberiu Tesileanu, Kyunghyun Cho, and Shirley Ho. Multiple physics pretraining for spatiotemporal surrogate models. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, editors, *Advances in Neural Information Processing Systems*, volume 37, pages 119301–119335. Curran Associates, Inc., 2024.

[38] Michael McCabe, Bruno Régaldo-Saint Blancard, Liam Holden Parker, Ruben Ohana, Miles Cranmer, Alberto Bietti, Michael Eickenberg, Siavash Golkar, Geraud Krawezik, Francois Lanusse, Mariel Pettee, Tiberiu Tesileanu, Kyunghyun Cho, and Shirley Ho. Multiple physics pretraining for physical surrogate models, 2024.

[39] Joschka Birk, Anna Hallin, and Gregor Kasieczka. Omnijet-α: the first cross-task foundation model for particle physics. *Machine Learning: Science and Technology*, 5(3):035031, aug 2024.

[40] Liu Yang, Siting Liu, Tingwei Meng, and Stanley J. Osher. In-context operator learning with data prompts for differential equation problems. *Proceedings of the National Academy of Sciences*, 120(39), September 2023.

[41] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[42] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. LLaMA: Open and efficient foundation language models, 2023.

[43] Meta AI. The llama 4 herd: The beginning of a new era of natively multimodal ai innovation, 2025. `https://ai.meta.com/blog/llama-4-multimodal-intelligence`.

[44] Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, et al. The pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*, 2020.

[45] Guilherme Penedo, Quentin Malartic, Daniel Hesslow, Ruxandra Cojocaru, Alessandro Cappelli, Hamza Alobeidli, Baptiste Pannier, Ebtesam Almazrouei, and Julien Launay. The refinedweb dataset for falcon llm: outperforming curated corpora with web data, and web data only. *arXiv preprint arXiv:2306.01116*, 2023.

[46] Junhao Zhao. LLMDataHub: Awesome Datasets for LLM Training, 2023. `https://github.com/Zjh-819/LLMDataHub`.

[47] Yang Liu, Jiahuan Cao, Chongyu Liu, Kai Ding, and Lianwen Jin. Datasets for large language models: A comprehensive survey, 2024.

[48] Christoph Schuhmann, Romain Beaumont, Richard Vencu, Cade Gordon, Ross Wightman, Mehdi Cherti, Theo Coombes, Aarush Katta, Clayton Mullis, Mitchell Wortsman, et al. Laion-5b: An open large-scale dataset for training next generation image-text models. *Advances in Neural Information Processing Systems*, 35:25278–25294, 2022.

[49] Haoping Bai, Shancong Mou, Tatiana Likhomanenko, Ramazan Gokberk Cinbis, Oncel Tuzel, Ping Huang, Jiulong Shan, Jianjun Shi, and Meng Cao. Vision datasets: A benchmark for vision-based industrial inspection, 2023.

[50] Tsai-Shien Chen, Aliaksandr Siarohin, Willi Menapace, Ekaterina Deyneka, Hsiang wei Chao, Byung Eun Jeon, Yuwei Fang, Hsin-Ying Lee, Jian Ren, Ming-Hsuan Yang, and Sergey Tulyakov. Panda-70m: Captioning 70m videos with multiple cross-modality teachers, 2024.

[51] Makoto Takamoto, Timothy Praditia, Raphael Leiteritz, Daniel MacKinlay, Francesco Alesiani, Dirk Pflüger, and Mathias Niepert. Pdebench: An extensive benchmark for scientific machine learning. *Advances in Neural Information Processing Systems*, 35:1596–1611, 2022.

[52] Jayesh K Gupta and Johannes Brandstetter. Towards multi-spatiotemporal-scale generalized pde modeling. *arXiv preprint arXiv:2209.15616*, 2022.

[53] Zhongkai Hao, Jiachen Yao, Chang Su, Hang Su, Ziao Wang, Fanzhi Lu, Zeyu Xia, Yichi Zhang, Songming Liu, Lu Lu, et al. PINNacle: A Comprehensive Benchmark of Physics-Informed Neural Networks for Solving PDEs. *arXiv preprint arXiv:2306.08827*, 2023.

[54] Georg Kohl, Li-Wei Chen, and Nils Thuerey. Benchmarking autoregressive conditional diffusion models for turbulent flow simulation. *arXiv*, 2023.

[55] Florent Bonnet, Jocelyn Mazari, Paola Cinnella, and Patrick Gallinari. Airfrans: High fidelity computational fluid dynamics dataset for approximating reynolds-averaged navier–stokes solutions. *Advances in Neural Information Processing Systems*, 35:23463–23478, 2022.

[56] Artur Toshev, Gianluca Galletti, Fabian Fritz, Stefan Adami, and Nikolaus Adams. Lagrangebench: A lagrangian fluid mechanics benchmarking suite. *Advances in Neural Information Processing Systems*, 36, 2024.

[57] Hans Hersbach, Bill Bell, Paul Berrisford, Shoji Hirahara, András Horányi, Joaquín Muñoz-Sabater, Julien Nicolas, Carole Peubey, Raluca Radu, Dinand Schepers, et al. The era5 global reanalysis. *Quarterly Journal of the Royal Meteorological Society*, 146(730):1999–2049, 2020.

[58] Sungduk Yu, Walter Hannah, Liran Peng, Jerry Lin, Mohamed Aziz Bhouri, Ritwik Gupta, Björn Lütjens, Justus Christopher Will, Gunnar Behrens, Julius Busecke, Nora Loose, Charles I Stern, Tom Beucler, Bryce Harrop, Benjamin R Hillman, Andrea Jenney, Savannah Ferretti, Nana Liu, Anima Anandkumar, Noah D Brenowitz, Veronika Eyring, Nicholas Geneva, Pierre Gentine, Stephan Mandt, Jaideep Pathak, Akshay Subramaniam, Carl Vondrick, Rose Yu, Laure Zanna, Tian Zheng, Ryan Abernathey, Fiaz Ahmed, David C Bader, Pierre Baldi, Elizabeth Barnes, Christopher Bretherton, Peter Caldwell, Wayne Chuang, Yilun Han, Yu Huang, Fernando Iglesias-Suarez, Sanket Jantre, Karthik Kashinath, Marat Khairoutdinov, Thorsten Kurth, Nicholas Lutsko, Po-Lun Ma, Griffin Mooers, J. David Neelin, David Randall, Sara Shamekh, Mark A Taylor, Nathan Urban, Janni Yuval, Guang Zhang, and Michael Pritchard. Climsim: A large multi-scale dataset for hybrid physics-ml climate emulation, 2024.

[59] Steeven Janny, Aurélien Benetteau, Madiha Nadri, Julie Digne, Nicolas Thome, and Christian Wolf. Eagle: Large-scale learning of turbulent fluid dynamics with mesh transformers. In *International Conference on Learning Representations (ICLR)*, 2023.

[60] Sheikh Md Shakeel Hassan, Arthur Feeney, Akash Dhruv, Jihoon Kim, Youngjoon Suh, Jaiyoung Ryu, Yoonjin Won, and Aparna Chandramowlishwaran. BubbleML: A multi-physics dataset and benchmarks for machine learning. In *Advances in Neural Information Processing Systems*, 2023.

[61] Wai Tong Chung, Bassem Akoush, Pushan Sharma, Alex Tamkin, Ki Sung Jung, Jacqueline H. Chen, Jack Guo, Davy Brouzet, Mohsen Talei, Bruno Savard, Alexei Y. Poludnenko, and Matthias Ihme. Turbulence in focus: Benchmarking scaling behavior of 3D volumetric super-resolution with BLASTNet 2.0 data. *Advances in Neural Information Processing Systems (NeurIPS)*, 36, 2023.

[62] Neil Ashton, Charles Mockett, Marian Fuchs, Louis Fliessbach, Hendrik Hetmann, Thilo Knacke, Norbert Schonwald, Vangelis Skaperdas, Grigoris Fotiadis, Astrid Walle, Burkhard Hupertz, and Danielle Maddix. Drivaerml: High-fidelity computational fluid dynamics dataset for road-car external aerodynamics, 2024.

[63] Ruben Ohana, Michael McCabe, Lucas Meyer, Rudy Morel, Fruzsina Agocs, Miguel Beneitez, Marsha Berger, Blakesly Burkhart, Stuart Dalziel, Drummond Fielding, et al. The well: a large-scale collection of diverse physics simulations for machine learning. *Advances in Neural Information Processing Systems*, 37:44989–45037, 2024.

[64] Fabien Casenave, Xavier Roynard, et al. PLAID: Physics Learning AI Datamodel. `https://gitlab.com/drti/plaid`, 2025.

[65] Marc Poinot and Christopher L Rumsey. Seven keys for practical understanding and use of cgns. In *2018 AIAA Aerospace Sciences Meeting*, page 1503, 2018.

[66] Quentin Lhoest, Albert Villanova del Moral, Patrick von Platen, Thomas Wolf, Mario Šaško, Yacine Jernite, Abhishek Thakur, Lewis Tunstall, Suraj Patil, Mariama Drame, Julien Chaumond, Julien Plu, Joe Davison, Simon Brandeis, Victor Sanh, Teven Le Scao, Kevin Canwen Xu, Nicolas Patry, Steven Liu, Angelina McMillan-Major, Philipp Schmid, Sylvain Gugger, Nathan Raw, Sylvain Lesage, Anton Lozhkov, Matthew Carrigan, Théo Matussière, Leandro von Werra, Lysandre Debut, Stas Bekman, and Clément Delangue. huggingface/datasets: 2.8.0, December 2022.

[67] Felipe Bordeu et al. MUSCAT: Mesh Utilities and Solver for Computational Analysis Toolkit. `https://gitlab.com/drti/muscat`, 2025.

[68] Felipe Bordeu, Fabien Casenave, and Julien Cortial. Basictools: a numerical simulation toolbox. *Journal of Open Source Software*, 8(86):5142, 2023.

[69] Fabien Casenave, Xavier Roynard, and Brian Staber. Tensile2d: 2D quasistatic non-linear structural mechanics solutions, under geometrical variations , February 2025.

[70] Mines ParisTech and ONERA the French aerospace lab. Zset: nonlinear material & structure analysis suite. http://www.zset-software.com, 1981-present.

[71] Brian Staber and Fabien Casenave. 2D_Multiscale_Hyperelasticity: a 2D quasistatic non-linear structural mechanics with finite elasticity and topology variations , February 2025.

[72] Igor A. Baratta, Joseph P. Dean, Jørgen S. Dokken, Michal Habera, Jack S. Hale, Chris N. Richardson, Marie E. Rognes, Matthew W. Scroggs, Nathan Sime, and Garth N. Wells. DOLFINx: the next generation FEniCS problem solving environment. preprint, 2023.

[73] Julien Yvonnet. *Computational homogenization of heterogeneous materials with finite elements*, volume 258. Springer, 2019.

[74] William PIAT and Fabien Casenave. 2D_ElastoPlastoDynamics: 2D dynamic non-linear structural mechanics dataset, with a non-linear non-local constitutive law , April 2025.

[75] OpenRadioss Community. Openradioss: Open-source finite element solver for dynamic event analysis, 2022. `https://openradioss.org/`.

[76] Xavier Roynard, Fabien Casenave, and Brian Staber. Rotor37: a 3D CFD RANS dataset, under geometrical variations of a compressor blade , February 2025.

[77] Cambier, Laurent, Heib, Sébastien, and Plot, Sylvie. The onera elsa cfd software: input from research and feedback from industry. *Mechanics & Industry*, 14(3):159–174, 2013.

[78] Fabien Casenave and Nissrine Akkari. 2D_profile: 2D external aero CFD RANS dataset, under geometrical variations , April 2025.

[79] Michele Alessandro Bucci, Luca Saverio, and Fabien Casenave. VKI-LS59: a 2D internal aero CFD RANS dataset, under geometrical variations , February 2025.

[80] Arthur Poulain, Cédric Content, Denis Sipp, Georgios Rigas, and Eric Garnier. Broadcast: A high-order compressible cfd toolbox for stability and sensitivity using algorithmic differentiation. *Computer Physics Communications*, 283:108557, 2023.

[81] Xavier Roynard, Fabien Casenave, and Brian Staber. AirfRANS_original, February 2025.

[82] Xavier Roynard, Fabien Casenave, and Brian Staber. AirfRANS_clipped, February 2025.

[83] Xavier Roynard, Fabien Casenave, and Brian Staber. AirfRANS_remeshed, February 2025.

[84] Mouadh Yagoubi, David Danan, Milad Leyli-Abadi, et al. ML4CFD Competition: Restrospective Analysis. *Work in progress*, 2025.

[85] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale, 2021.

[86] Ning Liu, Siavash Jafarzadeh, and Yue Yu. Domain Agnostic Fourier Neural Operators, 2023.

[87] Giovanni Catalani. MARIO: Multiscale Aerodynamic Resolution Invariant Operator, 2025. `https://github.com/giovannicatalani/MARIO`.

[88] Giovanni Catalani, Siddhant Agarwal, Xavier Bertrand, Frédéric Tost, Michael Bauerheim, and Joseph Morlier. Neural fields for rapid aircraft aerodynamics simulations. *Scientific Reports*, 14(1):25496, 2024.

[89] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks, 2017.

[90] Abbas Kabalan, Fabien Casenave, Felipe Bordeu, Virginie Ehrlacher, and Alexandre Ern. Morphing techniques for model order reduction with non parametric geometrical variabilities. In *16ème Colloque National en Calcul de Structures*, 2024.

[91] Rudy Geelen, Laura Balzano, and Karen Willcox. Learning latent representations in high-dimensional state spaces using polynomial manifold constructions. In *2023 62nd IEEE Conference on Decision and Control (CDC)*, pages 4960–4965. IEEE, 2023.

[92] Nicolas Barral, Tommaso Taddei, and Ishak Tifouti. Registration-based model reduction of parameterized PDEs with spatio-parameter adaptivity. *Journal of Computational Physics*, 499:112727, 2024.

[93] Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. *arXiv preprint arXiv:2001.04451*, 2020.

[94] Sinong Wang, Belinda Z Li, Madian Khabsa, Han Fang, and Hao Ma. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*, 2020.

[95] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale, 2021.

[96] Ashok Kumar Durairaj and Anandan Chinnalagu. Transformer based contextual model for sentiment analysis of customer reviews: A fine-tuned bert. *International Journal of Advanced Computer Science and Applications*, 12(11), 2021.

[97] Suhaima Jamal and Hayden Wimmer. An improved transformer-based model for detecting phishing, spam, and ham: A large language model approach, 2023.

[98] Xiang Dai, Ilias Chalkidis, Sune Darkner, and Desmond Elliott. Revisiting transformer-based models for long document classification, 2022.

[99] Enze Xie, Wenhai Wang, Zhiding Yu, Anima Anandkumar, Jose M. Alvarez, and Ping Luo. Segformer: Simple and efficient design for semantic segmentation with transformers, 2021.

[100] Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip Torr, and Vladlen Koltun. Point transformer, 2021.

[101] Jundou Jiang, Guanxiong Li, Yi Jiang, Laiping Zhang, and Xiaogang Deng. Transcfd: A transformer-based decoder for flow field prediction. *Engineering Applications of Artificial Intelligence*, 123:106340, 2023.

[102] Xiaoyang Wu, Li Jiang, Peng-Shuai Wang, Zhijian Liu, Xihui Liu, Yu Qiao, Wanli Ouyang, Tong He, and Hengshuang Zhao. Point transformer v3: Simpler, faster, stronger, 2024.

[103] George Karypis and Vipin Kumar. Metis – unstructured graph partitioning and sparse matrix ordering system, version 2.0. 01 1995.

[104] Luisa M Zintgraf, Kyriacos Shiarlis, Vitaly Kurin, Katja Hofmann, and Shimon Whiteson. Fast context adaptation via meta-learning, 2019.

A Details on the ML models used in the benchmark

We briefly present the main competing methods that we used for the benchmark. We also highlight some practical details about their implementation. Readers are encouraged to refer directly to the papers introducing the methods for further information.

A.1 MGN

A.1.1 Method

MeshGraphNet (MGN) [25], introduced by T. Pfaff et al., is a framework designed for learning mesh-based simulations using graph neural networks. The model is capable of being trained to simulate dynamic solutions by passing messages over a meshed domain, predicting acceleration at each mesh node at a given time step. This prediction allows for the calculation of the output field at the next time step through forward integration. Specifically, MGN is trained using one-step supervision and can be applied iteratively to generate long trajectories during inference. The architecture of MeshGraphNet is composed of encoding, processing, and decoding steps. In this work, MGN has been adapted to predict steady-state fields.

We utilize the following features as input (see Figure 5 for the workflow diagram):

- the distance of each node to the boundary,
- the type of node,
- the coordinates of the node.

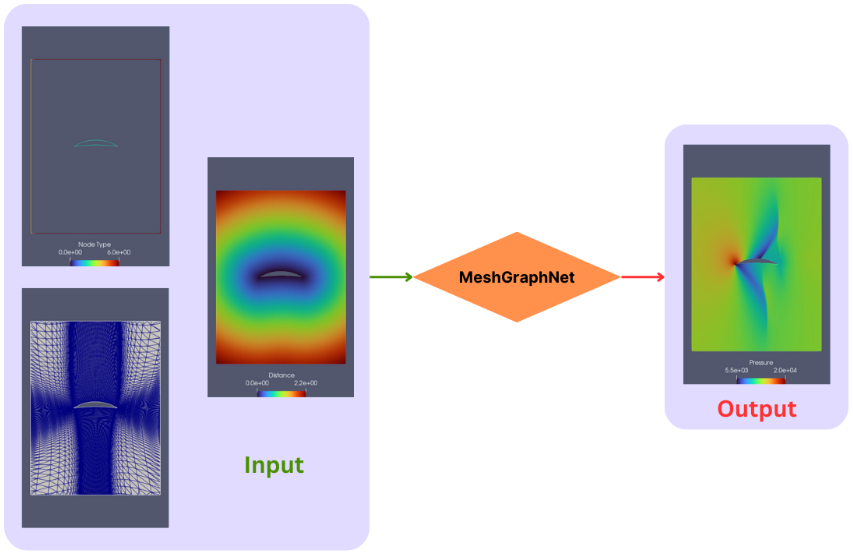


Figure 5: Illustration of MGN workflow to predict steady-state pressure field of a sample from the `2D_profile` dataset.

A.1.2 Experiments

In this section, we provide a summary of the experiments conducted on various datasets.

For all datasets, we trained two separate models: one focused on field predictions and the other on scalar predictions. For scalar outputs, a readout layer taken from [89] is added to the model. Except for the `2D_profile` dataset, we only required a single model since it does not include scalar prediction tasks.

The LeakyReLU is chosen as the activation function, and all models are trained for 1000 epochs.

The input node features consist of those introduced in the previous section, combined with input scalars if they exist. Given two node coordinates x_i and x_j, the calculation for edge features is based on $\exp(-||x_i - x_j||_2^2/(2h^2))$, where h epresents the median value of the edge lengths within the mesh.

The rest of architecture details and training information are outlined in Table 5 and Table 6.

`Dataset`	Message Passing Steps	Latent Size	Nbe epochs	Batch size	Training Time	Hardware
`Tensile2d`	10	16	1000	1	3h46min	$1 \times$ A100
`2D_MultiScHypEl`	10	32	1000	1	5h54min	$1 \times$ A100
`Rotor37`	10	64	1000	1	19h24min	$1 \times$ A100
`2D_profile`	10	128	1000	1	17h27min	$1 \times$ A100
`VKI-LS59`	10	64	1000	1	16h32min	$1 \times$ A100

Table 5: Field MGN: Architecture details and training statistics across datasets.

`Dataset`	Message Passing Steps	Latent Size	Nbe epochs	Batch size	Training Time	Hardware
`Tensile2d`	10	32	1000	1	4h6min	$1 \times$ A100
`2D_MultiScHypEl`	10	16	1000	1	6h	$1 \times$ A100
`Rotor37`	10	16	1000	1	10h	$1 \times$ A100
`VKI-LS59`	10	16	1000	1	9h13min	$1 \times$ A100

Table 6: Scalar MGN: Architecture details and training statistics across datasets.

A.2 MMGP

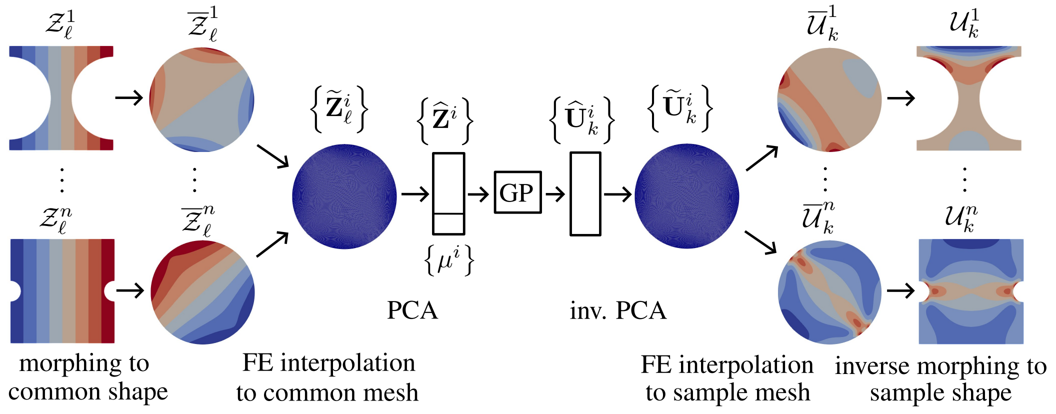
A.2.1 Method

We refer the reader to [19] for a complete presentation of the Mesh Morphing Gaussian Process (MMGP) method. MMGP combines four main ingredients: (i) mesh morphing, (ii) finite element interpolation, (iii) dimensionality reduction, and (iv) Gaussian process regression. Together, these enable learning mappings between geometries and solution fields for PDEs, even when the input geometry is provided as non-parametrized meshes.

An overview of the workflow is illustrated in Figure 6, which should be read from left to right. On the left are sample-specific input geometries; on the right are the corresponding solution fields defined on these geometries.

Since input meshes are not parametrized, they must first be embedded into a learnable space. MMGP does this by interpreting mesh vertex coordinates as continuous fields (e.g., the x-coordinate field shown in the left column of Figure 6, exhibiting vertical iso-lines). Each mesh is then deterministically morphed onto a reference geometry—the unit disk in this 2D example, but it can be one of the training samples shape. Next, each sample morphed coordinate fields are projected onto a common mesh of the reference geometry via finite element interpolation. This ensures all samples share a consistent discretization, making them compatible with standard dimensionality reduction techniques like PCA. The result is a compact, fixed-size representation of the geometry. When scalar inputs are present, they can be concatenated to the reduced vector.

A similar procedure is applied to the output fields: morphing onto the reference geometry, projection onto the common mesh, and PCA compression yield low-dimensional field representations aligned with the geometric embeddings.



$\mathcal{Z}_\ell^1$
$\overline{\mathcal{Z}}_\ell^1$
$\mathcal{Z}_\ell^n$
$\overline{\mathcal{Z}}_\ell^n$
$\{\widetilde{\mathbf{Z}}_\ell^i\}$
$\{\widehat{\mathbf{Z}}^i\}$
$\{\mu^i\}$
GP
$\{\widehat{\mathbf{U}}_k^i\}$
$\{\widetilde{\mathbf{U}}_k^i\}$
$\overline{\mathcal{U}}_k^1$
$\mathcal{U}_k^1$
$\overline{\mathcal{U}}_k^n$
$\mathcal{U}_k^n$
PCA
inv. PCA
morphing to common shape
FE interpolation to common mesh
FE interpolation to sample mesh
inverse morphing to sample shape

Figure 6: Illustration of the MMGP inference workflow for the prediction of an output field of interest [19].

These deterministic preprocessing steps transform the original complex problem—mapping between high-dimensional and irregularly discretized fields—into a standard regression task between low-dimensional vectors. This enables the use of classical regression models; we adopt Gaussian process regression due to its robustness, accuracy, and built-in uncertainty quantification.

MMGP offers several practical advantages: it handles large meshes efficiently, produces interpretable models, and delivers high accuracy in our experiments, with uncertainty estimates. In industrial design applications, where data can lie on low-dimensional manifolds, small models like MMGP can be especially effective—provided that the reparametrization (or embedding) is constructed appropriately, here with the morphing.

The main limitations of MMGP are tied to the morphing step, which currently requires problem-specific setup, and the fact that morphing and interpolation must still be performed at inference time. These challenges are addressed in recent works [20, 90], which introduce automatic alignment and online-efficient morphing strategies. Further improvements are proposed in [21], where optimization techniques are used to generate morphings that maximize PCA compression.

All mesh and field operations are implemented using the Muscat library [67, 68]. An upcoming release will include a GPU-accelerated finite element interpolation routine, significantly improving inference latency.

Additional improvement of MMGP are possible, by replacing the linear decoder of the PCA by a non-linear one that accounts for high-order interactions among the selected POD modes and includes a rotation of the POD basis and a polynomial correction, as proposed in [91].

Physics-based models compatible with the morphing, finite element interpolation and dimensionality reduction of MMGP have been proposed. The physics equation can be efficiently assembled and solved on the low-dimension space spanned by the PCA modes obtained after morphing, instead of using data-driven low-dimensional models. In [92], a hyper-reduced least-square Petrov-Galerkin scheme is used to reduced the Navier-Stokes equations, with morphing. While much more complicated to utilize, we expect such methods to greatly improve the accuracy, with a moderate additional computation cost.

A.2.2 Experiments

Hyperparameters and training statistics for the MMGP experiments are listed in Table 7. We first mention that MMGP has not been applied to the `2D_ElPlDynamics` and `2D_MultiScHypEl` datasets, since the method is yet to be extended to variable topology settings.

We notice that `Rotor37` and `VKI-LS59` do not require morphing, since the samples' meshes have the same number of nodes. In `Tensile2d` and `2D_profile`, systematic morphing strategies to align the shapes are sufficient, with respectively Tutte barycentric embedding [19, Ann B] and elasticity-based automatic morphing [20].

Since the `VKI-LS59` dataset exhibits discontinuities due to the presence of shock waves, a non-linear decoder [91] was employed to reconstruct the fields of interest. For the compression of the `mach` fields, 5 POD modes and a polynomial order of 3 were used, while 40 POD modes were retained for the compression of the `nut` fields. Since polynomial decoders are prone to overfitting, the number of modes and the polynomial order were selected through a k-fold cross-validation procedure on the training set.

Since the solution fields of `2D_profile` and `VKI-LS59` feature complex structures (e.g. shocks of variable position), we expect the involved optimal morphing strategy from [21] to significantly improve the results of MMGP on these cases.

Dataset	Morphing	PCA modes (shape)	PCA modes (field)	GP kernel	Training time	Hardware
`Tensile2d`	Tutte [19, Ann B]	8	8	Matérn 5/2	13min02s	128 cores
`Rotor37`	None	32	64	Matérn 5/2	6min13s	128 cores
`2D_profile`	Elasticity [20]	16	32	RBF	18min32s**	12 cores
`VKI-LS59`	None	13	5-3/40-1*	Matérn 5/2	4min13s	64 cores

Table 7: Hyperparameters and training statistics for the MMGP experiments (on an AMD EPYC 9534 CPU). Training times include all preprocessing (morphing, finite element interpolation and dimensionality reduction), in addition to the training of the Gaussian processes. *For `VKI-LS59`, X-Y stands for the number of modes and polynomial order of the decoder for the `mach` and `nut` fields respectively. **Not including morphing time (which takes approximately 10min on 300 cores).

A.3 Vi-Transformer and Augur

A.3.1 Method

Transformers for long context range regression. The natural way of dealing with mesh-based regression problems is to use GNN models which rely on message-passing. Although these are great at capturing information locally, they struggle to retrieve it at long distances. Indeed, the smallest number of GNN layers needed to have a receptive field that covers the whole graph is half the diameter of the graph. This becomes computationally impractical in the context of large simulation meshes. This behavior is analogous to Convolutional Neural Networks (CNNs) in Computer Vision (CV) where long-range dependencies are only captured at the deeper levels of the network. One way of alleviating this is to consider transformer architectures, which compute similarities between all the input tokens simultaneously thanks to the attention mechanism, thus removing the need to have infeasibly deep networks.

Transformers on very large data. One of the main challenges of transformers in this case is to handle the large size of the meshes (in the order of tens of thousands of points per mesh, and up to millions with practical industrial problems). Currently, the computational bottleneck of transformers is a widely considered subject: given N tokens of dimension D, the critical issue of self attention is that one needs $N^2 \times \mathcal{D}$ operations where $\mathcal{D} \approx D$ is the size of the embedding of each token, and N^2 is the cost to compute the Gram matrix of the N tokens (this computation cost is also a memory one as storing the matrix requires also $N^2\mathcal{D}$ numbers).

Many papers have focused on the possibility to linearize the cost of self-attention, for example:

- [93] introduces Reformer which considers the formulation of the attention mechanism : $\mathrm{softmax}\left(\frac{QK}{\sqrt{D}}\right)$ with the key and query matrices (respectively K and Q), capitalizing on the fact that for a given query Q_i, only the keys which provide high dot products with Q_i will have a significant impact on the value of $\mathrm{softmax}\left(\frac{Q_iK^T}{\sqrt{D}}\right)$. Therefore, Reformer makes use of locality-sensitive hashing for only computing the $Q_iK_j^T$ products with the p keys that are closest to a query, where $p \in \mathbb{N}$ is a chosen hyperparameter, efficiently linearizing the self attention.
- [94] introduces Linformer. Coarsely, Linformer relies on the Nyström approximation to approximate the Gram matrix of self attention. Precisely, while the Nyström approximation replaces an $n\times n$ symmetric matrix A by UU^T where U is only $n\times k$ containing the eigenvec-

tors of largest eigenvalue, Linformer offers to learn E, F such that $\mathrm{softmax}\left(\frac{QK}{\sqrt{D}}\right) \approx EF^T$. This also offers a linear approximation of the self attention computation.

This has also been tackled in CV tasks [95], where self-attention is not applied on pixels directly but on pixel-patches that aggregate pixel neighborhoods into tokens, thus drastically reducing the self-attention's input sequence length.

Transformers for large scale point-wise regression. The most used transformer architectures are in one of two categories. The auto-regressive sequence-to-sequence transformers, mostly used in Natural Language Processing (NLP) for text generation, and the sequence-to-class ones which are used both in NLP, as in sentiment analysis [96], spam detection [97], long document classification [98], and CV with image classification [95].

Both are quite different from the point-wise regression objective of the PLAID benchmarks. Indeed, the first method generates new token sequences of arbitrary lengths, while the second only makes use of transformer encoders with neural network heads to obtain a probability distribution on a set of classes.

Some work has been conducted in order to tackle regression problems with transformers:

- Segformer [99] addresses this in the case of image segmentation; it uses a multiscale U-type transformer to sequentially downscale the input image, and uses a multiscale MLP head to decode these downscaled states into the output segmentation mask.
- Point Transformer [100] also uses a U-style encoder-decoder architecture, this time on 3D point-cloud data for both segmentation and classification.
- TransCFD [101] tackles airfoil surrogate CFD modeling by using a decoder-only architecture from a latent embedding of the input geometry. It relies on structured regular grids (images) of the inputs, and not arbitrary mesh discretizations.
- Point Transformer V3 [102] groups points together and computes attention scores within these groups. Local and long-distance information are captured through different serializations of the input mesh.

Both Segformer and TransCFD make use of the regular nature of their data to precisely decode (and/or encode) the output (and/or input) fields. Point Transformers, on the other hand, handle unstructured point-cloud data. Although these methods fit the nature of the PLAID benchmark, we propose lighter methods that stick more closely to the classical transformer model.

Vi-Transformer for mesh field regression. The chosen approach relies on a transformer encoder architecture and is analogous to Vision Transformers (Vi-Transformer). Rather than considering each node of the mesh as a token by its own, the encoder takes as input tokenized point-cloud patches. Local information is kept within the patches while long-range information is retrieved through the transformer's mapping, which compares all token pairs together. The general architecture of the Vi-Transformer is depicted in Figure 7.

Augur Transformer model. Augur has developed Transformer models specifically designed for numerical simulations. These models share fundamental architectural similarities with Vision Transformers (ViT), where the computational mesh is decomposed into patches. Each patch is embedded into a latent space, resulting in the input tokens for the Transformer architecture. This approach enables information exchange between local patches across long spatial distances, similar to how ViTs process image data.

The key innovation in Augur's approach lies in the decoding mechanism, addressing a critical question: how to properly reconstruct the output field from the processed sequence of tokens? In traditional ViT architectures, direct reconstruction from individually processed tokens can result in discontinuities at patch interfaces due to insufficient global context integration. Augur models overcome this limitation by incorporating a global information vector that aggregates data from all tokens. The decoder then uses a combination of point-specific information, processed local features, and global context to produce a more robust and consistent output field. Furthermore, unlike ViTs, Augur models do not treat scalar predictions as constant fields but instead derive them directly from

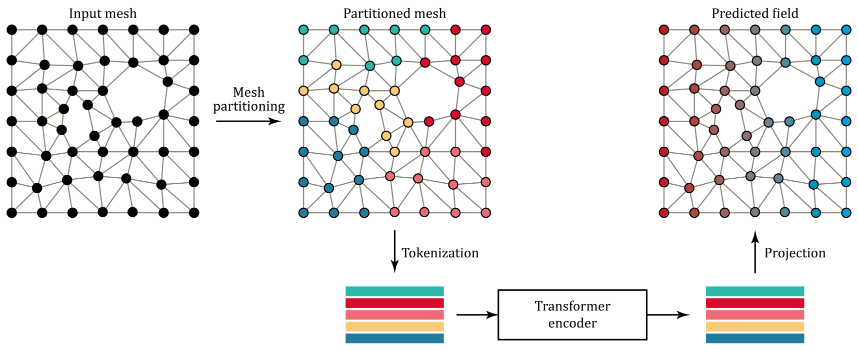


Figure 7: Vi-Transformer architecture. Input meshes are partitioned using the Metis domain decomposition algorithm [103]. Each such sub-domain is then tokenized before passing through the transformer encoder. In the end, each token is decoded into its domain's corresponding fields. Input scalars are embedded during the tokenization procedure while output scalars are estimated as uniform fields.

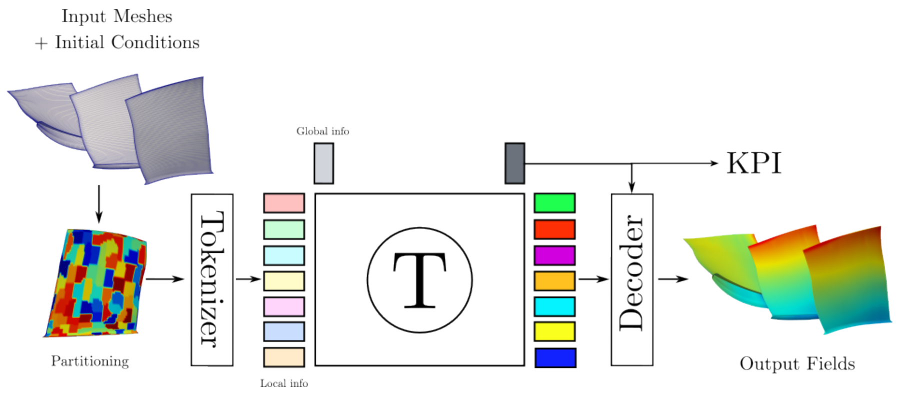


Figure 8: Augur Transformer architecture: Input meshes are partitioned using the Metis domain decomposition algorithm. Each subdomain is then tokenized before being passed through the Transformer. An additional global tensor is added to the Transformer to gather global information. Output fields are reconstructed using a decoder that leverages both local and global information. Output scalars (KPIs) are predicted directly from the global tensor.

the global information vector, enhancing prediction accuracy. The general architecture of the Augur model is depicted in Figure 8.

A.3.2 Experiments

Both the Vi-Transformer and Augur models rely on a relatively small number of hyperparameters. These include the patch size (i.e., the number of nodes per patch), the latent dimension onto which the aggregated patches are projected, and the Transformer encoder hyperparameters, such as the number of heads, the number of transformer encoder layers and the dimension of the feedforward layer. Table 8 details the hyperparameters for the Vi-Transformer, while Table 9 outlines those for the Augur model.

A.4 DAFNO

DAFNO belongs to the Operator Approximator class of architectures, i.e. it builds mappings between two function spaces. The use of the Fast Fourier Transform (FFT) within the different layers leads to the sampling of the input function on a regular grid, thus falling back to a finite dimension space. This architecture has the advantage of learning transformation in the frequency domain which provides a significant advantage compared to CNNs on several physical problems.

Dataset	Patch size	Latent dimension	Feedforward dimension	Nb encoder layers	Training time	Hardware
Tensile2d	50	6400	2048	5	3h18	3 × A30
2D_MultiScHypEl	10	512	256	5	1h56	3 × A30
Rotor37	100	256	256	10	33min	3 × A30
2D_profile	50	1024	1024	5	36min	3 × A30
VKI-LS59	50	6400	2048	5	1h27	3 × A30

Table 8: (Vi-Transformer) Hyperparameters and training statistics for the Vi-Transformer experiments. Training times include all preprocessing (domain decomposition, tokenization), in addition to the training of the model itself. The number of attention heads is kept at 16 for all experiments.

Dataset	Patch size	Latent dimension	Feedforward dimension	Nb encoder layers	Training time	Hardware
Tensile2d	16	512	2048	8	1h11	1 × RTX 2080Ti
2D_MultiScHypEl	4	128	512	8	7h48	1 × RTX 2080Ti
Rotor37	32	256	1024	8	2h30	1 × RTX 2080Ti
VKI-LS59	64	512	2048	4	2h15	1 × RTX 2080Ti

Table 9: (Augur) Hyperparameters and training statistics for the Augur experiments. Training times include all preprocessing (domain decomposition, tokenization), in addition to the training of the model itself.

A.4.1 Method

The DAFNO model deals separately with the input fields and the geometry of the problem [86]: let $u : \mathbb{R}^2 \to \mathbb{R}^k$ be our input fields and $\chi_\Omega : \mathbb{R}^2 \to \{0,1\}$ be the characteristic function of the domain Ω. Let $W \in \mathbb{R}^{k\times k}$, $W^* \in \mathbb{R}^{k\times k}$, $c \in \mathbb{R}^k$ be the learnable parameters, let $\sigma : \mathbb{R} \to \mathbb{R}$ be a scalar non-linear function (sigmoid, ReLU, or tanH) to be applied elementwise. A layer of the DAFNO architecture is defined by the following operator:

$$\begin{aligned}\mathcal{J}[u](x) =&\ \sigma\left(Wu(x) + c + \mathcal{F}^{-1}\left[W^*\mathcal{F}\left[\left(u(x) - u(\cdot)\right)\chi_\Omega(\cdot)\chi_\Omega(x)\right]\right](x)\right) && (1)\\ =&\ \sigma\left(Wu(x) + c + \chi_\Omega(x)\mathcal{I}\left[\chi_\Omega(\cdot)u(\cdot)\right](x) - u(x)\mathcal{I}\left[\chi_\Omega(\cdot)\right](x)\right), && (2)\end{aligned}$$

where $\mathcal{I}[f](x) = \mathcal{F}^{-1}\left[W^*\mathcal{F}[u](.)\right](x)$, with $\mathcal{F}$ denoting the FFT operator. Equation (1) shows the interest of using the DAFNO architecture: the FFT in operator only considers values inside the domain Ω. Moreover the FFT is computed over the local variation of the input field rather than the input field itself ($u(x) - u(.)$ instead of $u(.)$) making the layer, by design, seek features within local variations. The DAFNO network ends up being a composition of one or multiple of such layers. The mask χ_Ω is used at each layer unaltered to make sure that no noise outside the domain may perturb the prediction.

FNO models and variant can only predict on regular grids (this is due to the use of the FFT). This is a common constraint shared with some neural networks such as CNNs. This means that, in order to predict on an unstructured mesh, a preprocessing and postprocessing of the fields are needed. The preprocessing consists in a projection of the original mesh to a regular grid where the FNO is able produce a prediction. Then, a postprocessing projecting back from the regular grid to the original mesh needs to be performed to compare the prediction to reference fields. The projection operations were performed using Muscat [68, 67].

A.4.2 Experiments

The DAFNO architecture can build transient predictions on various geometries and topology, the only dataset introduced in Section 4 that meets these three characteristics is the `2D_ElPlDynamics` dataset.

Training procedure. The training was performed in a autoregressive manner: given the input fields at time t, the model has to predict the fields at time $t + dt$ very much like an explicit solver would do.

Once trained, one may build the whole transient field prediction by applying the model recursively on the initial conditions. A key choice involves selecting inputs that are informative enough for the model to accurately predict the system dynamics.

On top of the fields provided by the dataset (`U_x` and `U_y`) we added two coordinate fields (one for x and one for y) and we computed a fifth input: a smoothed mask $\chi_\Omega^{\text{smooth}}$ as suggested by the original DAFNO paper [86] along with being a drop in replacement of χ_Ω in the DAFNO layers. This smooth quantity is richer than its discontinuous counterpart since it provides insight on how close we are from the border of the geometry. We are summarizing the input/output quantities in Table 10.

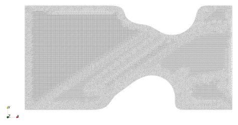
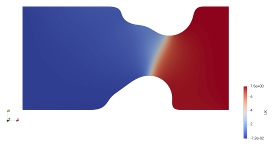
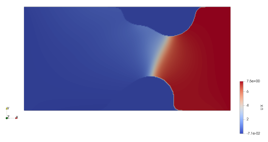
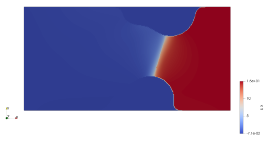
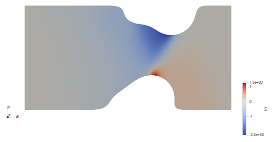
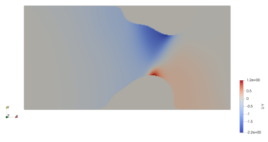
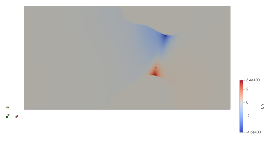
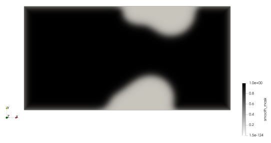
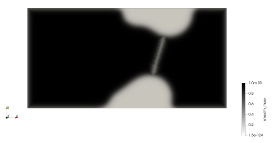
Attribute	Simulation at t	Input DAFNO	Output DAFNO
Mesh		-	-
`U_x`			
`U_y`			
$\chi_\Omega^{\text{smooth}}$	-		

Table 10: Features throughout the learning process, "-" means the field is not available/used at the given stage. The simulation at t (column 1) can be projected to a regular grid (column 2). The regular fields along with coordinates fields x and y make input features for the DAFNO model which in turns predicts the fields at $t + dt$ (column 3).

The training was parallelized on 40 GPUs (A100) and lasted 6 hours. Inference and thus testing can be performed on a single GPU to compute the metrics presented in Table 4.

Model and training parametrization. We summarize the model parametrization in Table 11 and training procedure in Table 12.

Model parameters	Layer count	Channel hidden layers	Padding	Fourier modes	Activation Function
Value	8	64	8	20×20	GELU

Table 11: DAFNO: parametrization of the model.

A.5 MARIO

Modulated Aerodynamic Resolution-Invariant Operator (MARIO) is a deep learning model designed to approximate the solution operator of a partial differential equation (PDE) [87], involving geometric variability. It leverages Conditional Neural Fields (or Implicit Neural Representations) to learn

Training parameters	Epochs	Optimizer	Learning rate	Batch size	Loss
Value	1800	Adam	0.0003	60	Pixel-wise L_2

Table 12: DAFNO: parametrization of the training.

the mapping between spatial coordinates from a mesh, geometric information (e.g., via the signed distance function, SDF), inflow conditions, and the resulting physical field. Unlike mesh-based methods, INRs represent continuous fields through neural network parameterizations, enabling resolution-independent predictions and flexible evaluation. MARIO extends this approach to handle multiple geometries and operating conditions through a conditioning mechanism.

A.5.1 Method

Modulated INR architecture. MARIO implements a conditional neural field approach where a single neural network architecture can represent multiple distinct signals through a conditioning mechanism. The conditioning variable $z = [\mu_{\text{geom}}, \mu]$ encodes both geometric parameterization μ_{geom} and operating conditions μ (e.g., angle of attack, Mach number, Reynolds number).

The main network is a multilayer perceptron (MLP) where the layer outputs are modulated by sample-specific vectors:

$$f_{\theta,\phi}(x) = W_L(\eta_{L-1} \circ \eta_{L-2} \circ \cdots \circ \eta_1 \circ \gamma(x)) + b_L \tag{3}$$
$$\eta_l(\cdot) = \text{ReLU}(W_l(\cdot) + b_l + \phi_l(z)) \tag{4}$$

where $\phi_l(z) = [h_\psi(z)]_l \in \mathbb{R}^{d_l}$ are layer-specific modulation vectors obtained from the hypernetwork h_ψ that processes the conditioning variable z. The main network parameters θ are shared for all samples and consist of the weights and biases matrices W_l, b_l. In MARIO, an explicit shape encoding μ_{geom} is used as input of the architecture to properly model geometric variability. In many real-world applications, a geometric parameterization is not available or insufficient to capture complex shapes. Therefore, a learning mechanism to obtain compact geometric representations from the SDF fields is adopted. These encoding process leverages a separate Neural Field encoder, that maps input coordinates to output SDF values, while fitting latent shape representations.

Geometry encoding mechanism. For each geometry's signed distance function (SDF), a meta-learning optimization procedure based on CAVIA [104] adapts a shared neural network $f_{\theta_{in},\phi_{in}}$ to represent different shapes. Given the shared network parameters θ_{in} and hypernetwork parameters ψ, the latent representation $\mu_{geom} = z_{in}^{(K)}$ for geometry i is obtained by solving:

$$z_{in}^{(0)} = 0 \tag{5}$$
$$z_{in}^{(k+1)} = z_{in}^{(k)} - \alpha \nabla_{z_{in}^{(k)}} \mathcal{L}_{in}(f_{\theta_{in},\phi_{in}}(x), sdf_i), \quad \text{for } 0 \le k \le K-1 \tag{6}$$

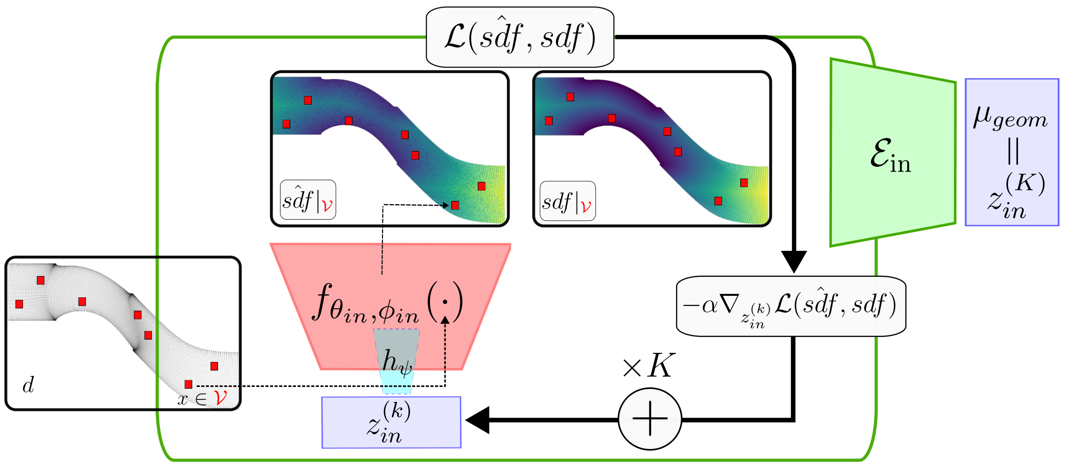
where $\phi_{in} = h_\psi(z_{in}^{(k)})$, α is the inner loop learning rate, and K is the number of optimization steps (typically set to 3). The loss $\mathcal{L}_{in}$ measures the reconstruction error between the true SDF field and its prediction over a sampling grid defined on the input domain.

This optimization process, illustrated in Figure 9, yields a compact latent code $\mu_{\text{geom}} = z_{in}^{(K)}$ that captures the essential geometric features.

Fourier feature encoding. To address the spectral bias inherent in neural networks, MARIO employs Fourier feature encoding for the input coordinates:

$$\gamma(x) = [\cos(2\pi \mathbf{B}x), \sin(2\pi \mathbf{B}x)] \tag{7}$$

where $\mathbf{B} \in \mathbb{R}^{m \times d}$ contains frequency vectors sampled from a Gaussian distribution $\mathcal{N}(0, \sigma)$. This encoding enables the network to better capture high-frequency details in the output fields.



$\mathcal{L}(\hat{sdf}, sdf)$
$\hat{sdf}|_{\mathcal{V}}$
$sdf|_{\mathcal{V}}$
$\mathcal{E}_{\text{in}}$
$\mu_{geom} \,||\, z_{in}^{(K)}$
$f_{\theta_{in},\phi_{in}}(\cdot)$
h_ψ
$-\alpha \nabla_{z_{in}^{(k)}} \mathcal{L}(\hat{sdf}, sdf)$
$\times K$
d
$x \in \mathcal{V}$
$z_{in}^{(k)}$

Figure 9: MARIO geometry encoding process.

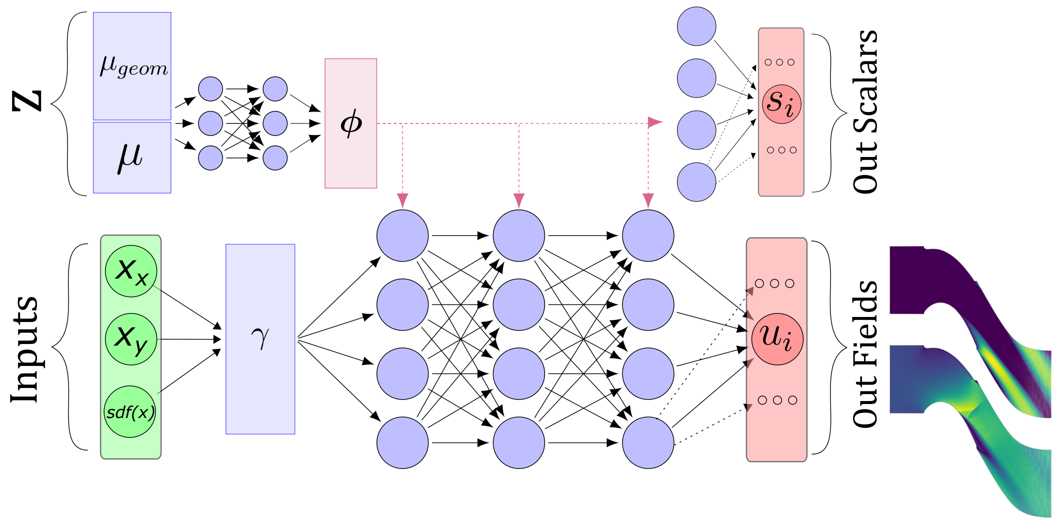


Figure 10: MARIO architecture.

Scalar output prediction. In addition to predicting coordinate-dependent fields, MARIO can also predict global scalar quantities for each sample. Since these scalar outputs are global properties of the solution (e.g., power coefficients, efficiency metrics), they depend only on the sample-specific information encoded in the modulation vectors. The scalar prediction is therefore implemented as:

$$s = W_s \cdot \phi_{agg} + b_s \tag{8}$$

where ϕ_{agg} represents an aggregation of the modulation vectors produced by the hypernetwork. This single-layer transformation efficiently leverages the already learned sample representation without requiring additional feature extraction.

The architecture of MARIO is illustration in Figure 10.

Training procedure. MARIO is trained using a weighted loss function that balances field prediction accuracy and scalar output accuracy:

$$\mathcal{L} = \alpha \cdot \mathcal{L}_{\text{field}} + (1 - \alpha) \cdot \mathcal{L}_{\text{scalar}} \tag{9}$$

817 where $\alpha \in [0, 1]$ is a weighting parameter. The field loss $\mathcal{L}_{\text{field}}$ is computed as the mean squared error between predicted and target fields across spatial locations, while the scalar loss $\mathcal{L}_{\text{scalar}}$ is the mean squared error of the global quantities.

Key advantages. MARIO presents three major benefits: (i) it is resolution-invariant and can be evaluated at arbitrary spatial locations; (ii) it overcomes spectral bias through multiscale Fourier encodings; and (iii) it adapts to geometry-specific variations via bias modulation using the auxiliary network h_ψ.

A.5.2 Experiment

Model and training parametrization. The model parametrization and training procedure are provided respectively in Tables 13 and 14.

Model param.	Geom. Hypernet. depth	Geom. Hypernet. width	Geom. latent dim	Hypernet. depth	Hypernet. width	INR depth	INR width	Nb of frequencies
Value	1	128	16	3	256	6	256	64

Table 13: MARIO: parametrization of the model.

Training param.	Epochs	Optimizer	Learning rate	Batch size	Training time	Training hardware	Loss ($\alpha = 0.8$)
Value	2000	AdamW	0.001	4	30h	$1 \times$ A100	MSE

Table 14: MARIO: parametrization of the training.

We notice that MARIO is significantly longer to train than the other tested models.

B Additional details on PLAID

We illustrate further the capabilities of PLAID by providing some additional commands to retrieve information from our datasets directly from Hugging Face.

B.1 `Tensile2d`

`Tensile2d` is a simple dataset, for which standard and simple PLAID commands are sufficient to retrieve the data:

```python
from datasets import load_dataset
from plaid.containers.sample import Sample
import pickle

# Load the dataset
hf_dataset = load_dataset("PLAID-datasets/Tensile2d", split="all_samples")

# Get split ids
ids_train = hf_dataset.description["split"]["train_500"]

# Get inputs/outputs names
in_scalars_names = hf_dataset.description["in_scalars_names"]
out_fields_names = hf_dataset.description["out_fields_names"]

# Get samples
sample = Sample.model_validate(pickle.loads(hf_dataset[ids_train[0]]["sample"]))

# Examples of data retrievals
nodes = sample.get_nodes()
elements = sample.get_elements()
```

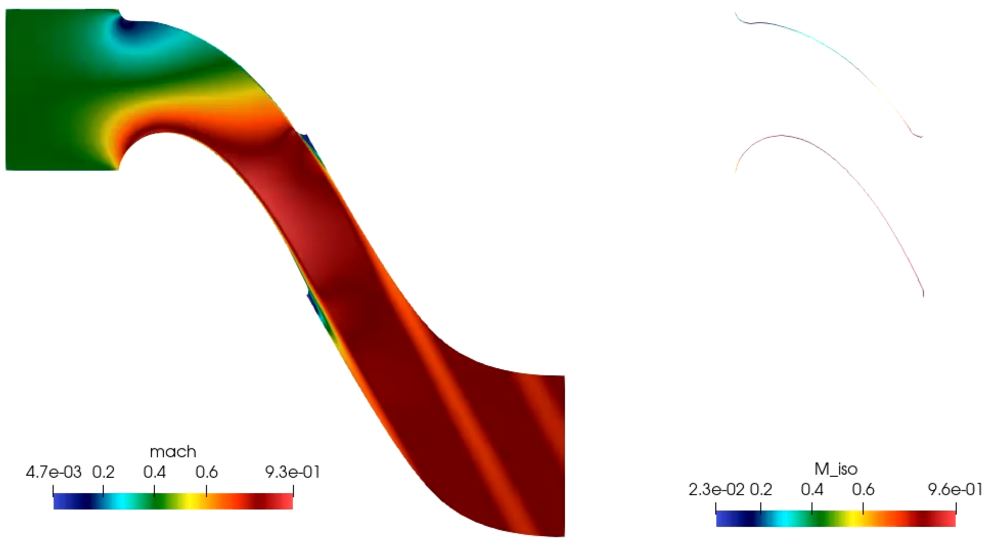


Figure 11: Illustration of the first sample in the train split of `VKI-LS59`: (left) fluid domain, (right) blade surface domain.

```python
nodal_tags = sample.get_nodal_tags()

for sn in ["P", "p1", "p2", "p3", "p4", "p5"]:
    scalar = sample.get_scalar(sn)

# outputs
for fn in ["U1", "U2", "q", "sig11", "sig22", "sig12"]:
    field = sample.get_field(fn)

for sn in ["max_von_mises", "max_q", "max_U2_top", "max_sig22_top"]:
    scalar = sample.get_scalar(sn)

```

The geometrical support in PLAID samples can be easily converted to Muscat meshes:

```python
from Muscat.Bridges import CGNSBridge
CGNS_tree = sample.get_mesh()
mesh = CGNSBridge.CGNSToMesh(CGNS_tree)
```

B.2 `VKI-LS59`

`VKI-LS59` also contains stationary configurations, meaning only one time step per sample, but features a complex geometrical setting, with a 2D fluid domain and a 1D blade surface domain, see Figure 11.

The fluid domain contains 2D elements in a 2D ambient space, hence is contained in the CGNS base called "Base_2_2". For the blade surface domain, we have 1D elements in a 2D ambient space: the CGNS base is then "Base_1_2". The corresponding data are retrieved as follows:

```python
from datasets import load_dataset
from plaid.containers.sample import Sample
import pickle

# Load the first sample of the train split
hf_dataset = load_dataset("PLAID-datasets/VKI-LS59", split="all_samples")
```

```python
ids_train = hf_dataset.description["split"]["train"]
sample = Sample.model_validate(pickle.loads(hf_dataset[ids_train[0]]["sample"]))

# Examples of data retrievals
for fn in ["sdf", "ro", "rou", "rov", "roe", "nut", "mach"]:
    field = sample.get_field(fn, base_name="Base_2_2")
M_iso = sample.get_field("M_iso", base_name="Base_1_2")
for sn in sample.get_scalar_names():
    scalar = sample.get_scalar(sn)

nodes_fluid = sample.get_nodes(base_name="Base_2_2")
nodes_blade_surface = sample.get_nodes(base_name="Base_1_2")
elements_fluid = sample.get_elements(base_name="Base_2_2")
elements_blade_surface = sample.get_elements(base_name="Base_1_2")
nodal_tag_fluid = sample.get_nodal_tags(base_name="Base_2_2")
```

The meshes for the fluid domain and blade surface domain can also be converted to Muscat meshes:

```python
from Muscat.Bridges import CGNSBridge
CGNS_tree = sample.get_mesh()
mesh_fluid = CGNSBridge.CGNSToMesh(CGNS_tree, baseNames=["Base_2_2"])
mesh_blade = CGNSBridge.CGNSToMesh(CGNS_tree, baseNames=["Base_1_2"])
```

B.3 `2D_ElPlDynamics`

`2D_ElPlDynamics` contains additional complexity: time-dependent data and a field located at the center of the elements. When retrieving data, the default location of the fields is at the vertices. For other type of fields, location mush be specified. Furthermore, in `2D_ElPlDynamics`, the mesh is different from one sample to another, but stays constant through the time sequence within a sample. Hence, to prevent useless duplication of data, we link the geometrical support of the second to last time step data to the mesh of the first time step. The corresponding commands are provided below:

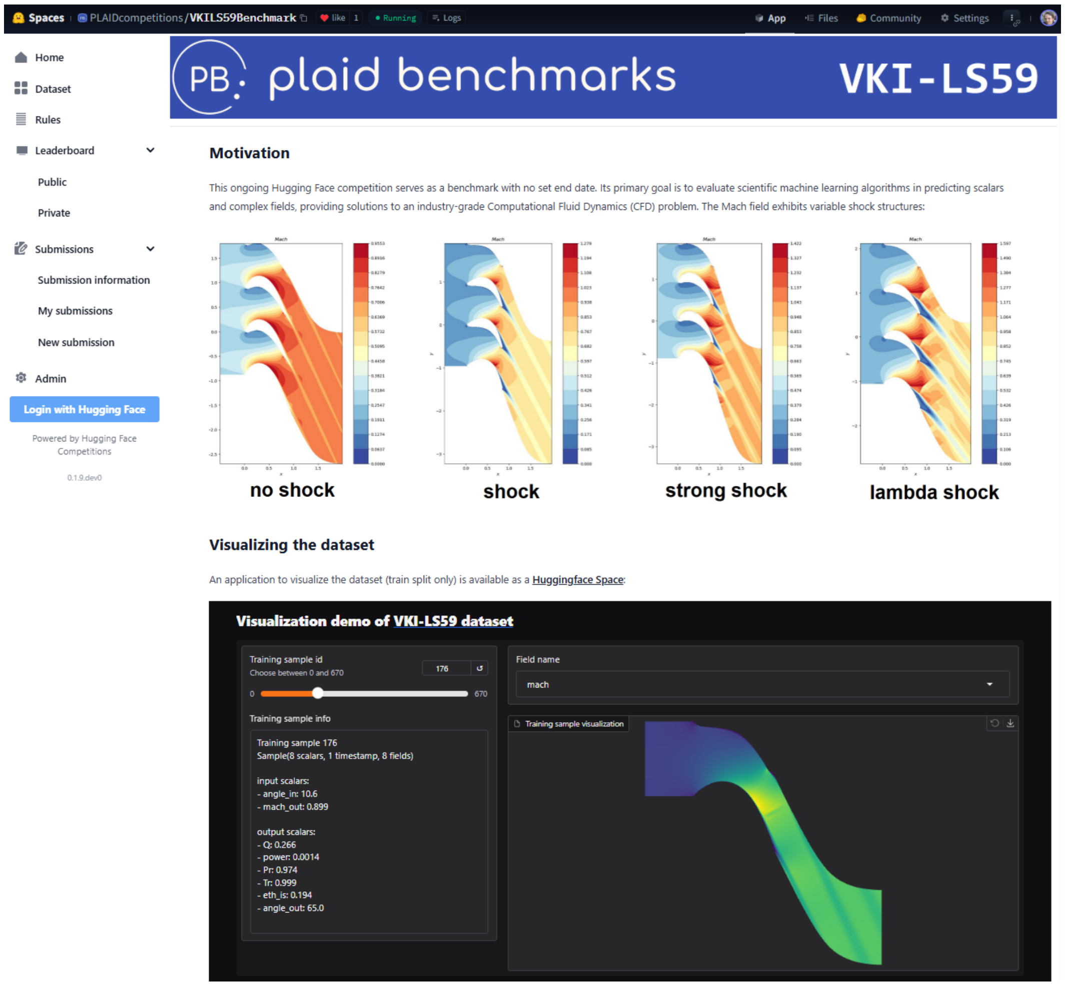
```python
from datasets import load_dataset
from plaid.containers.sample import Sample
import pickle

# Load the first sample of the train split
hf_dataset = load_dataset("PLAID-datasets/2D_ElastoPlastoDynamics",
↪  split="all_samples")
ids_train = hf_dataset.description["split"]["train"]
sample = Sample.model_validate(pickle.loads(hf_dataset[ids_train[0]]["sample"]))

# Examples of data retrievals
time_steps = sample.get_all_mesh_times()

for time in time_steps:
    for fn in ["U_x","U_y"]:
        field = sample.get_field(fn, time = time)
    field = sample.get_field("EROSION_STATUS", location="CellCenter", time = time)

CGNS_tree_t0 = sample.get_mesh(time = 0.)
CGNS_tree_t1 = sample.get_mesh(time = 0.01, apply_links = True, in_memory = True)
```

C Benchmarking online applications

Anyone wishing to participate in our benchmarks, hosted at huggingface.co/PLAIDcompetitions, should create a Hugging Face account. However, no account is required to browse the website or view the leaderboards. To participate, users simply need to train their model independently and submit predictions on the testing set. We do not require participants to upload their models. Two separate leaderboards are maintained, each based on a hidden subset of the test set, in order to discourage tentatives to overfit on the testing set.

We illustrate the benchmarking application using the `VKI-LS59` dataset as an example.

Figure 12 shows the benchmark homepage. A navigation menu is available on the left-hand side, allowing users to browse the site and log in. This page also provides examples of the dataset output fields and includes a visualization tool, where users can select a training sample ID and an output field to display.



Motivation

This ongoing Hugging Face competition serves as a benchmark with no set end date. Its primary goal is to evaluate scientific machine learning algorithms in predicting scalars and complex fields, providing solutions to an industry-grade Computational Fluid Dynamics (CFD) problem. The Mach field exhibits variable shock structures:

Visualizing the dataset

An application to visualize the dataset (train split only) is available as a Huggingface Space:

Visualization demo of VKI-LS59 dataset

Training sample info

```
Training sample 176
Sample(8 scalars, 1 timestamp, 8 fields)

input scalars:
- angle_in: 10.6
- mach_out: 0.899

output scalars:
- Q: 0.266
- power: 0.0014
- Pr: 0.974
- Tr: 0.999
- eth_is: 0.194
- angle_out: 65.0
```

Figure 12: "Home" page of the benchmarking application on the `VKI-LS59` dataset.

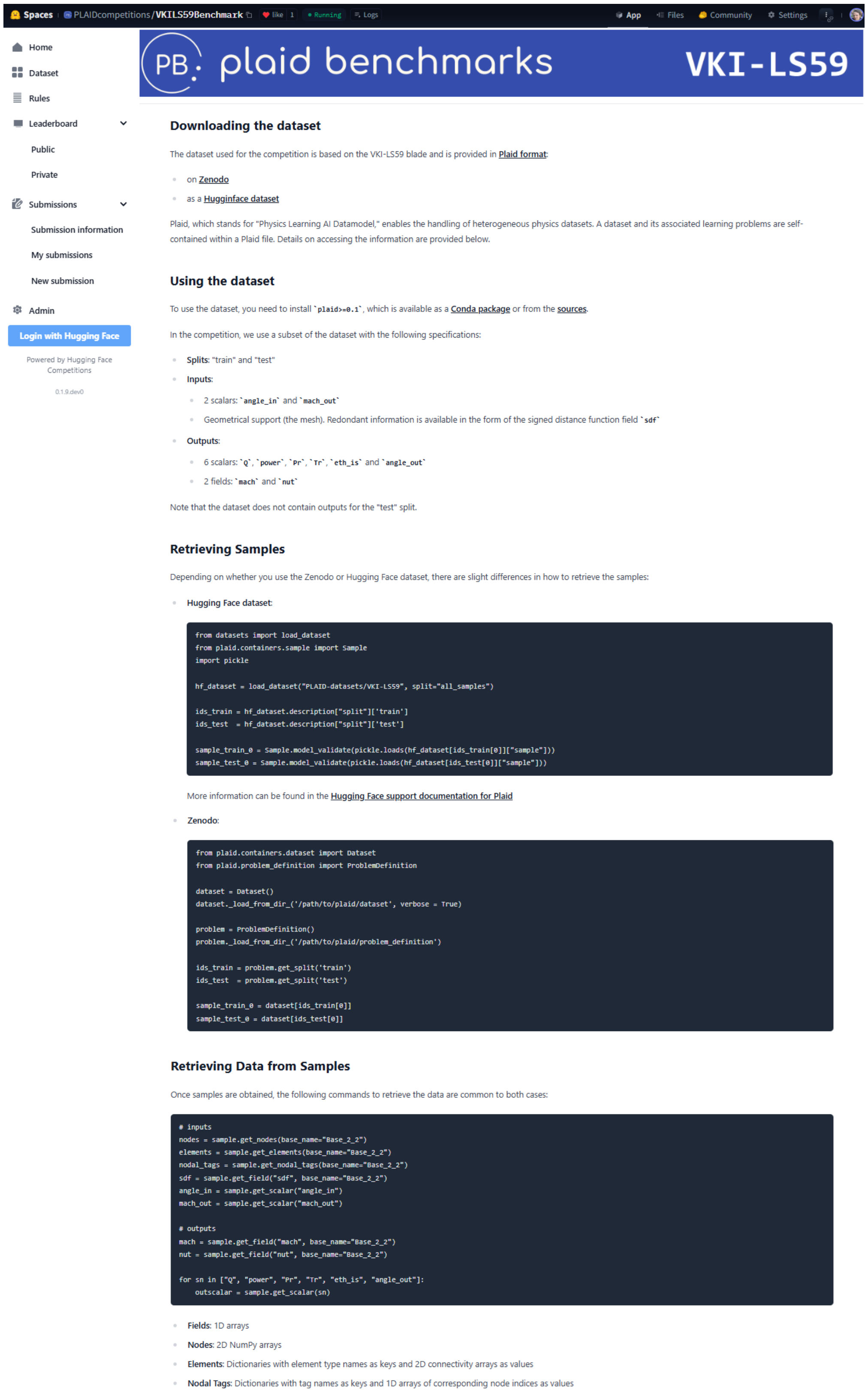
Figure 13 provides detailed instructions on how to retrieve the dataset, including a description of the inputs and outputs used in the benchmark. Example commands are also provided to retrieve the samples and the required associated data.

The set of rules applying to the benchmark is presented in Figure 14.

Figure 15 provides detailed instructions on how to generate and submit the prediction file. The scoring function used for evaluation is also described.

Figure 16 illustrates the user's submissions page and the submission interface.

Figure 17 shows the public leaderboard as it appeared at the time of submission of this work.



Downloading the dataset

The dataset used for the competition is based on the VKI-LS59 blade and is provided in Plaid format:

- on Zenodo
- as a Hugginface dataset

Plaid, which stands for "Physics Learning AI Datamodel," enables the handling of heterogeneous physics datasets. A dataset and its associated learning problems are self-contained within a Plaid file. Details on accessing the information are provided below.

Using the dataset

To use the dataset, you need to install `` `plaid>=0.1` ``, which is available as a Conda package or from the sources.

In the competition, we use a subset of the dataset with the following specifications:

- **Splits**: "train" and "test"
- **Inputs**:
  - 2 scalars: `` `angle_in` `` and `` `mach_out` ``
  - Geometrical support (the mesh). Redondant information is available in the form of the signed distance function field `` `sdf` ``
- **Outputs**:
  - 6 scalars: `` `Q` ``, `` `power` ``, `` `Pr` ``, `` `Tr` ``, `` `eth_is` `` and `` `angle_out` ``
  - 2 fields: `` `mach` `` and `` `nut` ``

Note that the dataset does not contain outputs for the "test" split.

Retrieving Samples

Depending on whether you use the Zenodo or Hugging Face dataset, there are slight differences in how to retrieve the samples:

- **Hugging Face dataset**:

```
from datasets import load_dataset
from plaid.containers.sample import Sample
import pickle

hf_dataset = load_dataset("PLAID-datasets/VKI-LS59", split="all_samples")

ids_train = hf_dataset.description["split"]['train']
ids_test  = hf_dataset.description["split"]['test']

sample_train_0 = Sample.model_validate(pickle.loads(hf_dataset[ids_train[0]]["sample"]))
sample_test_0 = Sample.model_validate(pickle.loads(hf_dataset[ids_test[0]]["sample"]))
```

  More information can be found in the Hugging Face support documentation for Plaid

- **Zenodo**:

```
from plaid.containers.dataset import Dataset
from plaid.problem_definition import ProblemDefinition

dataset = Dataset()
dataset._load_from_dir_('/path/to/plaid/dataset', verbose = True)

problem = ProblemDefinition()
problem._load_from_dir_('/path/to/plaid/problem_definition')

ids_train = problem.get_split('train')
ids_test  = problem.get_split('test')

sample_train_0 = dataset[ids_train[0]]
sample_test_0 = dataset[ids_test[0]]
```

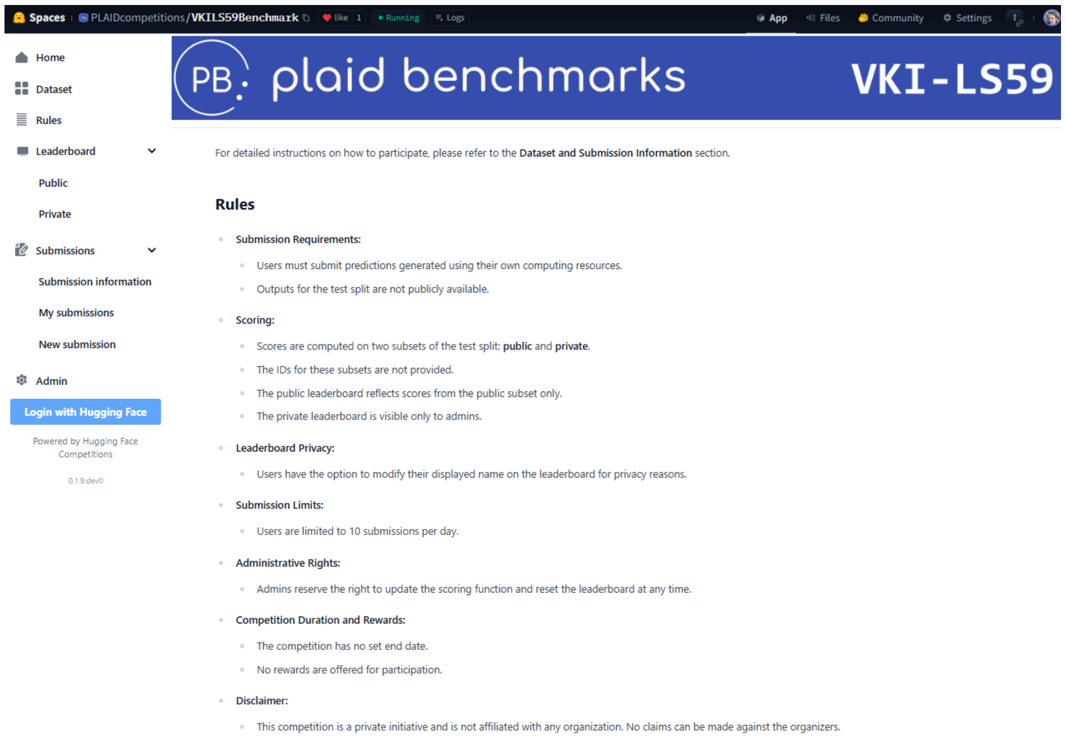
Retrieving Data from Samples

Once samples are obtained, the following commands to retrieve the data are common to both cases:

```
# inputs
nodes = sample.get_nodes(base_name="Base_2_2")
elements = sample.get_elements(base_name="Base_2_2")
nodal_tags = sample.get_nodal_tags(base_name="Base_2_2")
sdf = sample.get_field("sdf", base_name="Base_2_2")
angle_in = sample.get_scalar("angle_in")
mach_out = sample.get_scalar("mach_out")

# outputs
mach = sample.get_field("mach", base_name="Base_2_2")
nut = sample.get_field("nut", base_name="Base_2_2")

for sn in ["Q", "power", "Pr", "Tr", "eth_is", "angle_out"]:
    outscalar = sample.get_scalar(sn)
```

- **Fields**: 1D arrays
- **Nodes**: 2D NumPy arrays
- **Elements**: Dictionaries with element type names as keys and 2D connectivity arrays as values
- **Nodal Tags**: Dictionaries with tag names as keys and 1D arrays of corresponding node indices as values

Figure 13: "Dataset" page of the benchmarking application on the VKI-LS59 dataset.

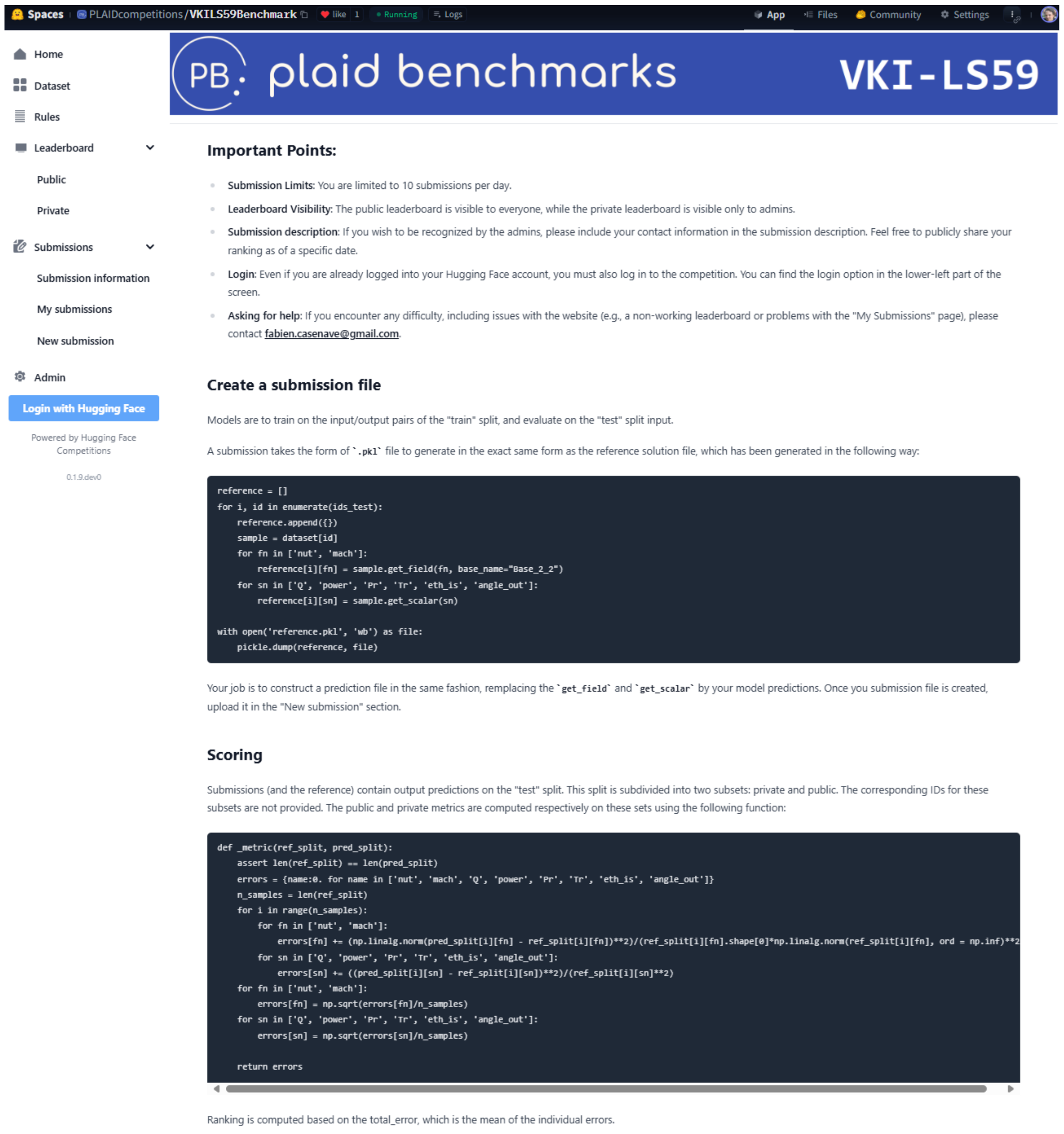


For detailed instructions on how to participate, please refer to the **Dataset and Submission Information** section.

Rules

- **Submission Requirements:**
  - Users must submit predictions generated using their own computing resources.
  - Outputs for the test split are not publicly available.
- **Scoring:**
  - Scores are computed on two subsets of the test split: **public** and **private**.
  - The IDs for these subsets are not provided.
  - The public leaderboard reflects scores from the public subset only.
  - The private leaderboard is visible only to admins.
- **Leaderboard Privacy:**
  - Users have the option to modify their displayed name on the leaderboard for privacy reasons.
- **Submission Limits:**
  - Users are limited to 10 submissions per day.
- **Administrative Rights:**
  - Admins reserve the right to update the scoring function and reset the leaderboard at any time.
- **Competition Duration and Rewards:**
  - The competition has no set end date.
  - No rewards are offered for participation.
- **Disclaimer:**
  - This competition is a private initiative and is not affiliated with any organization. No claims can be made against the organizers.

Figure 14: "Rules" page of the benchmarking application on the `VKI-LS59` dataset.



Important Points:

- **Submission Limits**: You are limited to 10 submissions per day.
- **Leaderboard Visibility**: The public leaderboard is visible to everyone, while the private leaderboard is visible only to admins.
- **Submission description**: If you wish to be recognized by the admins, please include your contact information in the submission description. Feel free to publicly share your ranking as of a specific date.
- **Login**: Even if you are already logged into your Hugging Face account, you must also log in to the competition. You can find the login option in the lower-left part of the screen.
- **Asking for help**: If you encounter any difficulty, including issues with the website (e.g., a non-working leaderboard or problems with the "My Submissions" page), please contact fabien.casenave@gmail.com.

Create a submission file

Models are to train on the input/output pairs of the "train" split, and evaluate on the "test" split input.

A submission takes the form of `` `.pkl` `` file to generate in the exact same form as the reference solution file, which has been generated in the following way:

```
reference = []
for i, id in enumerate(ids_test):
    reference.append({})
    sample = dataset[id]
    for fn in ['nut', 'mach']:
        reference[i][fn] = sample.get_field(fn, base_name="Base_2_2")
    for sn in ['Q', 'power', 'Pr', 'Tr', 'eth_is', 'angle_out']:
        reference[i][sn] = sample.get_scalar(sn)

with open('reference.pkl', 'wb') as file:
    pickle.dump(reference, file)
```

Your job is to construct a prediction file in the same fashion, remplacing the `` `get_field` `` and `` `get_scalar` `` by your model predictions. Once you submission file is created, upload it in the "New submission" section.

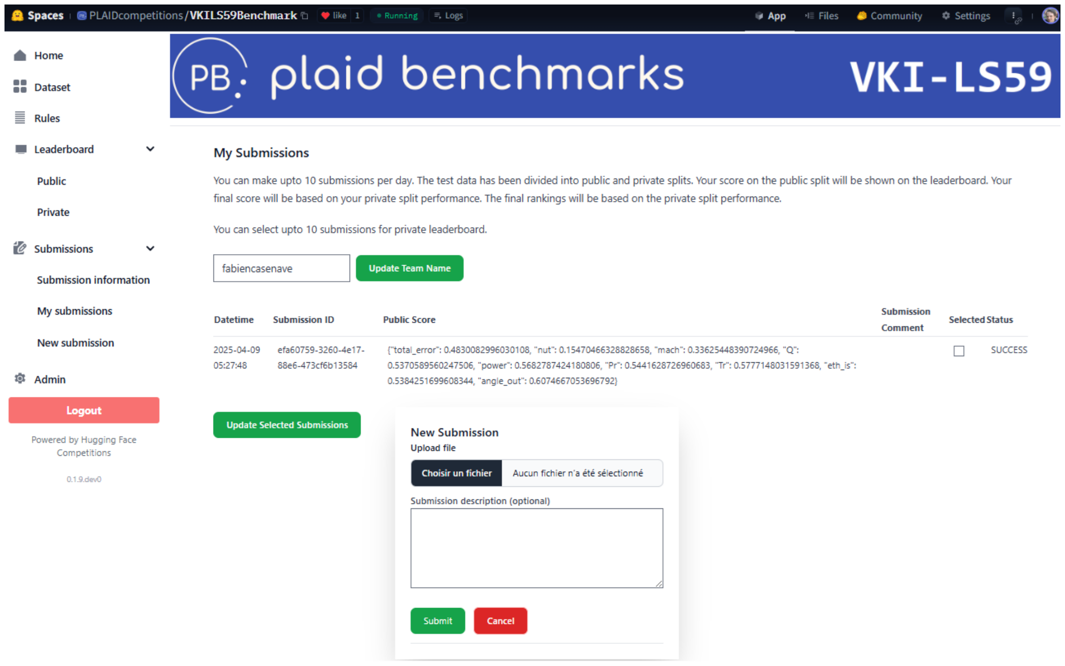
Scoring

Submissions (and the reference) contain output predictions on the "test" split. This split is subdivided into two subsets: private and public. The corresponding IDs for these subsets are not provided. The public and private metrics are computed respectively on these sets using the following function:

```
def _metric(ref_split, pred_split):
    assert len(ref_split) == len(pred_split)
    errors = {name:0. for name in ['nut', 'mach', 'Q', 'power', 'Pr', 'Tr', 'eth_is', 'angle_out']}
    n_samples = len(ref_split)
    for i in range(n_samples):
        for fn in ['nut', 'mach']:
            errors[fn] += (np.linalg.norm(pred_split[i][fn] - ref_split[i][fn])**2)/(ref_split[i][fn].shape[0]*np.linalg.norm(ref_split[i][fn], ord = np.inf)**2
        for sn in ['Q', 'power', 'Pr', 'Tr', 'eth_is', 'angle_out']:
            errors[sn] += ((pred_split[i][sn] - ref_split[i][sn])**2)/(ref_split[i][sn]**2)
    for fn in ['nut', 'mach']:
        errors[fn] = np.sqrt(errors[fn]/n_samples)
    for sn in ['Q', 'power', 'Pr', 'Tr', 'eth_is', 'angle_out']:
        errors[sn] = np.sqrt(errors[sn]/n_samples)

    return errors
```

Ranking is computed based on the total_error, which is the mean of the individual errors.

Figure 15: "Submission information" page of the benchmarking online application on the VKI-LS59 dataset.



My Submissions

You can make upto 10 submissions per day. The test data has been divided into public and private splits. Your score on the public split will be shown on the leaderboard. Your final score will be based on your private split performance. The final rankings will be based on the private split performance.

You can select upto 10 submissions for private leaderboard.

fabiencasenave | Update Team Name

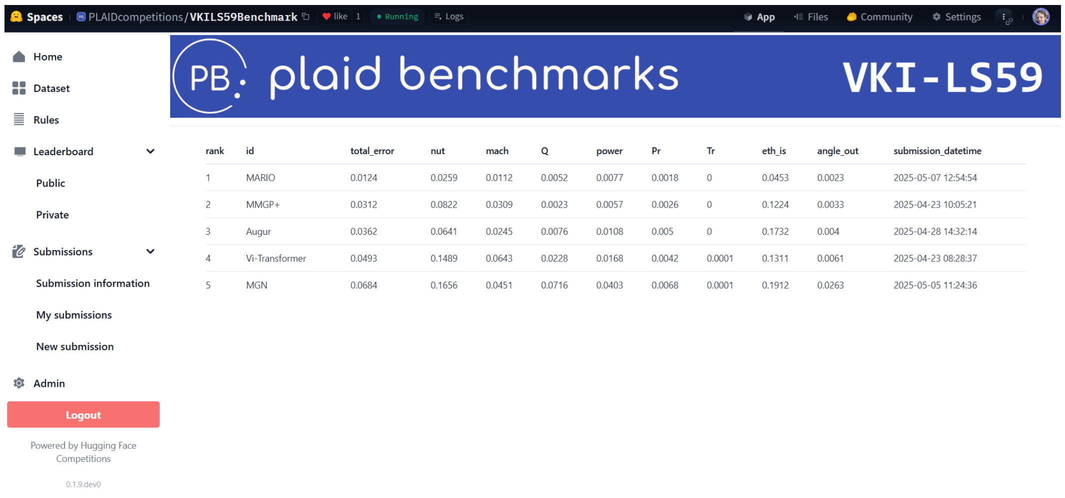
Datetime	Submission ID	Public Score	Submission Comment	Selected	Status
2025-04-09 05:27:48	efa60759-3260-4e17-88e6-473cf6b13584	{"total_error": 0.4830082996030108, "nut": 0.15470466328828658, "mach": 0.33625448390724966, "Q": 0.5370589560247506, "power": 0.5682787424180806, "Pr": 0.5441628726960683, "Tr": 0.5777148031591368, "eth_is": 0.5384251699608344, "angle_out": 0.6074667053696792}		☐	SUCCESS

Update Selected Submissions

New Submission
Upload file
Choisir un fichier | Aucun fichier n'a été sélectionné
Submission description (optional)

Submit | Cancel

Figure 16: "My submissions" page of the benchmarking application on the `VKI-LS59` dataset.



rank	id	total_error	nut	mach	Q	power	Pr	Tr	eth_is	angle_out	submission_datetime
1	MARIO	0.0124	0.0259	0.0112	0.0052	0.0077	0.0018	0	0.0453	0.0023	2025-05-07 12:54:54
2	MMGP+	0.0312	0.0822	0.0309	0.0023	0.0057	0.0026	0	0.1224	0.0033	2025-04-23 10:05:21
3	Augur	0.0362	0.0641	0.0245	0.0076	0.0108	0.005	0	0.1732	0.004	2025-04-28 14:32:14
4	Vi-Transformer	0.0493	0.1489	0.0643	0.0228	0.0168	0.0042	0.0001	0.1311	0.0061	2025-04-23 08:28:37
5	MGN	0.0684	0.1656	0.0451	0.0716	0.0403	0.0068	0.0001	0.1912	0.0263	2025-05-05 11:24:36

Figure 17: "Public leaderboard" page of the benchmarking application on the `VKI-LS59` dataset.

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: Claims are made in the abstract and the last two paragraphs on the introduction. Section 3, 4 and 5 directly address the claimed contributions, namely the proposed standard and implementation, the datasets and the benchmarking results and tools.

   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: In the conclusion (Section 6), we mention that PLAID may not currently address all possible future complex scenario, but can be adapted. We also infer that the dataset collection and benchmarks can be completed by new data and methods, and provide a future roadmap for PLAID. We specify in Section 5.3 that the benchmark results are not exhaustive.

   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 include theoretical results.

   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: PLAID: the code is provided and extensively documented, including tutorials for converting anyone how data into the PLAID data model; instructions and advice are given for anyone to download and inspect the datasets. Benchmarking: applications and detailed instructions are provided for anyone to participate and update the benchmarks.

   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: The PLAID library is provided in open-source and documented. All the mentioned datasets are provided in open-data. Benchmarking application are provided and available for anyone to use and contribute to.

   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: Learning tasks are formalized in the PLAID standard, and information on split and inputs/outputs are provided in Section 4. Details on the models used in the benchmarks are provided in Section A.

   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: [No]

Justification: While our contributions focuse on the PLAID standard and implementation, the datasets and the benchmarking, it would have been very computationally demanding to train each model of the benchmark many times to include mean and standard deviations for every experiment. Besides, ranking in the benchmark is made from a single prediction.

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 are provided in Section A.

   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: The NeurIPS Code of Ethics has been read and acknowledged by the first author. In particular, the research did not involve real data of real people, is not concerned by safety, security, discrimnation, surveillance, deception & harassment, environment, human rights, bias and fairness. Impact mitigation measures are 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: [NA]

    Justification: The paper deals with ML for physics, with main application for design processes in the industry.

    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: The paper deals with ML for physics, with no apparent risk of misuse.

    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: Origin and license of the `AirfRANS` dataset is correctly reported (origin in the paper, origin and license in the Zenodo and Hugging Face repos). The rest of the propose material is original, the chosen license for PLAID and the datasets is provided in the repos.

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: The PLAID library, the datasets and the benchmarking application are all described and documented in details.

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: The paper does not involve crowdsourcing nor 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: The paper does not involve crowdsourcing nor 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: [NA]

Justification: The core method development in this research does not involve LLMs as any important, original, or non-standard components.

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.