Evaluating Spatiotemporal Prediction Models in a Low-Data Regime.

Andrzej Dulny ✉, Paul Heinisch, Andreas Hotho, and Anna Krause

University of Würzburg, Germany
{andrzej.dulny,andreas.hotho,anna.krause}@uni-wuerzburg.de
paul.heinisch@stud-mail.uni-heidelberg.de

Abstract. Predicting the evolution of physical systems governed by PDEs from sparse, irregular observations remains a major challenge. While most data-driven methods focus on dense, grid-structured data, real-world applications often involve limited, noisy measurements. We extend the DynaBench benchmark to evaluate fourteen modern models—including neural operators and graph-based methods—across multiple PDEs, spatial resolutions, and observation patterns. Our study reveals how sparsity and spatial structure affect model performance in low-data regimes. All code and resources are publicly released to support reproducibility and future work.

Keywords: small data · low resolution · PDE · benchmark

1 Introduction

Understanding and predicting the temporal evolution of physical systems governed by partial differential equations (PDEs) is a central challenge at the intersection of machine learning and computational physics. Accurate modeling is essential in domains such as climate and weather forecasting [27,26,16], fluid dynamics [21,35,9], and material science [3,10]. Traditional numerical solvers can approximate solutions with dense initial and boundary conditions, but real-world applications often rely on sparse, noisy observations at irregular, non-grid locations [6]. Most data-driven approaches still target high-resolution, grid-structured datasets with uniformly sampled inputs [15,18,27].

In contrast, relatively little attention has been paid to the low-data regime, particularly when observations are both spatially sparse and irregularly distributed. In some domains—most notably weather and climate modeling—graph neural networks have shown promise in forecasting physical systems from densely sampled data across arbitrary spatial locations [16]. However, only a few recent studies have addressed challenges from sparse observations [6,22], and these generally lack systematic evaluation of diverse model classes across different observation densities and spatial configurations. The DynaBench benchmark [6] provides a valuable initial step, evaluating several deep learning models for six physical systems in a low-resolution setting, comparing grid-based and non-uniform observation methods. Yet, the benchmark is limited to a small set of architectures,

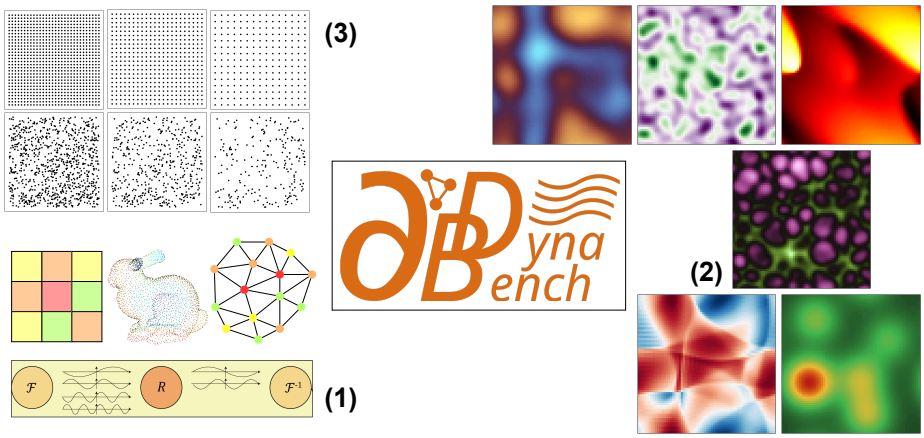

Fig. 1: We evaluate fourteen different deep learning models suited for forecasting spatiotemporal physical systems **(1)** on six distinct physical systems described by partial differential equations **(2)** using four different resolutions (only three depicted) and two distinct patterns (grid vs non-grid) of observation points **(3)**. We use the DynaBench benchmark suit for our experiments.

most notably not including neural operators, and assesses performance at only one spatial resolution. Significant questions remain about how model performance varies across data regimes and observation patterns.

In this work, we present a benchmark study evaluating modern machine learning models' ability to capture the temporal evolution of physical systems from limited, irregularly spaced observations. We focus on scenarios where dynamics are governed by well-known PDEs—such as the Burgers', wave, and advection equations—while data is spatially sparse and irregularly sampled. To this end, we use the DynaBench dataset, extending evaluation to four spatial resolutions and two spatial structures (grid and non-grid). Using this setup, we systematically assess the performance of fourteen diverse models on the task of predicting physical system dynamics under challenging low-data conditions.

Our contributions can be summarized as follows:

1. We extend the existing DynaBench benchmark by evaluating physical dynamics learning across multiple PDEs, resolutions, and spatial structures with sparse, irregular observations.
2. We evaluate fourteen state-of-the-art models, including neural operators and graph-based methods, on low-data forecasting tasks.
3. We analyze how observation sparsity and spatial configuration affect model performance and robustness.
4. We release all code[1], evaluation tools[2], and model implementations to support reproducibility and future research.

[1] Available at `https://professor-x.de/low_data_benchmark`
[2] Dataset available at `https://dynabench.github.io/`

2 Related Work

Machine learning methods have become increasingly popular for modeling physical systems, particularly those governed by partial differential equations (PDEs). These approaches offer promising alternatives or complements to traditional numerical solvers, especially in scenarios with noisy observations or partially known physics. In response, several benchmarking efforts have emerged to standardize the evaluation of deep learning models in scientific and spatiotemporal forecasting tasks.

PDEBench [30] is a widely used benchmark for learning time-dependent PDEs from data. It includes a broad suite of equations and simulation settings, providing high-resolution datasets generated via numerical solvers. However, it assumes full access to grid-level fields and is therefore unsuitable for sparse or irregular data scenarios. LagrangeBench [31] is a physics simulation benchmark, which focuses on Lagrangian particle systems rather than field-based PDE modeling and emphasizes high-resolution simulations rather than sparse measurement settings. CFDBench [24] addresses generalization in fluid dynamics by evaluating models across varying PDE parameters, boundary conditions, and geometries. While valuable for understanding robustness to physical variations, CFDBench also relies on high-resolution grid-like features and is not designed for sparse or irregularly sampled input data. Graph Mesh Benchmark [2] provides steady-state airflow simulations over airfoils, formatted as graphs derived from CFD meshes. Although it highlights the utility of geometric deep learning for mesh-based simulations, its focus is on steady-state prediction under fixed conditions, with limited relevance to time-evolving systems or sparse dynamic observations. Large-Scale Fluid Benchmark [12] targets incompressible Navier-Stokes simulations. However, it primarily supports evaluating discretization schemes and lacks the diversity, scale, or sparse data settings needed for training modern ML surrogates under real-world constraints. In contrast to these high-resolution benchmarks, DynaBench [6] is the only existing benchmark specifically designed for evaluating model performance in the low-data regime. It features six PDE-based systems and emphasizes forecasting from sparse, irregularly distributed observations without access to full grid-level information. While DynaBench includes graph-based and point-cloud models suited for this task, it evaluates only a limited set of architectures and does so only at a single spatial resolution.

Machine learning approaches to PDEs can be broadly categorized into two types: models that solve individual PDE instances, and operator learning methods that learn mappings between function spaces. Our work focuses on the latter, targeting the learning of solution operators that map the state at a given time t to the state of the physical system at state $t + 1$. Several neural operator architectures have been developed, differing in how they represent or discretize function spaces. Key examples include the DeepONet [23], which uses different representations for value and location spaces (branch and trunc networks); the Fourier Neural Operator (FNO) [18], which performs calculations in Fourier space using FFT, requiring functions on a uniform grid in real space; and Graph Neural Operators (GNOs [19] and MGNOs [20]), which use pointwise values

as nodes of graphs. A comprehensive review by Kovachki et al. [15] compares various neural operator architectures and highlights the strong performance of the FNO in a range of PDE systems. Based on these findings, we consider the FNO as our primary operator learning model for the benchmark. A key limitation of the original FNO is its reliance on uniformly spaced input data, which restricts its applicability in settings with irregular spatial sampling. To address this issue, Lingsch et al. [22] introduce the non-uniform Fourier Neural Operator (GeoFNO), a variant that extends the FNO to non-uniform input grids by substituting FFT with a direct calculation of the Fourier transform. GeoFNO achieves comparable accuracy to the original FNO on structured domains while offering improved computational efficiency and flexibility in handling irregular geometries.

3 Benchmark Setup

In this section, we describe the experimental setup for benchmarking the selected architectures on the Dynabench dataset [6]. We include the architectures included in the original Dynabench benchmark [6] and extend them with impactful models from the field of predictive learning and learning from sparse data in recent years.

We consider models that allow learning from point cloud structured data, as well as models that require a 2D grid structure. Models that are classified as cloud or graph models do not require the grid structure and use the coordinates and associated features at these points as input. Graph neural networks require more structure in the form of edges between points. The edges may be predefined or chosen to connect certain nodes during runtime, e.g., using a KNN algorithm. An overview of the models with their respective domains can be seen in Table 1.

3.1 Dynabench Dataset

The Dynabench dataset consists of six equations in four resolutions in either a grid or cloud structure. The six PDEs in the dataset are the Advection (**A**), Burgers' (**B**), Gas-Dynamics (**GD**), Kuramoto–Sivashinsky (**KS**), Reaction-Diffusion (**RD**), and Wave (**W**) equations. The four resolutions are comprised of *low* with $K = 225$ points, *medium* with $K = 484$ points, *high* with $K = 900$ points, and *full* with $K = 4096$ points. Note that the square root of the individual number of points (15, 22, 30 and 64 respectively) determines the grid size in the case of grid-structured data. The full resolution contains the original points that were used during the numerical simulation of each equation and, therefore, is only available in the grid structure. All other resolutions are a subset of the *full* resolution points. In the case of grid data, the sampled points are again aligned in a grid; for the cloud structure, the points are sampled randomly. The locations are consistent within one evolution trajectory.

For models working on arbitrary structured data (point cloud and graph) we additionally transform each grid dataset to a cloud structure, by flattening the

Table 1: Evaluated models with the respective domain the model expects and the type of architecture. The domain is the type of data that can be used for training the respective model. Grid models require a uniform grid of observations, cloud models expect the data to be in a data structure with features and coordinates for each point, and graph models require an underlying graph structure. Models in bold have not been included in the original Dynabench benchmark [6].

Model	Domain	Architecure	Citation	Year
CNN	grid	grid	Lecun, Y. et al. [17]	1998
ResNet	grid	grid	He, K. et al. [11]	2015
FNO	grid	grid	Li, Z. et al. [18]	2020
NeuralPDE	grid	grid	Dulny, A. et al. [5]	2022
GrINd	arbitrary	cloud	Dulny, A., et al. [4]	2024
PTv1	arbitrary	cloud	Zhao, H. et al. [37]	2021
PTv3	arbitrary	cloud	Wu, X. et al. [34]	2024
GeoFNO	arbitrary	cloud	Lingsch, L. et al. [22]	2023
PointNet	arbitrary	graph	Qi, C.R. et al. [28]	2017
PointGNN	arbitrary	graph	Shi, W. et al. [29]	2020
GraphPDE	arbitrary	graph	Iakovlev, V. et al. [13]	2021
GAT	arbitrary	graph	Veličković, P. et al. [32]	2018
GCN	arbitrary	graph	Kipf, T. et al. [14]	2017
FeaStNet	arbitrary	graph	Verma, N. et al. [33]	2018

grid. This allows directly controlling and evaluating the influence of the underlying data structure versus the model architecture. Additionally, for all models using cloud structured data we calculate the graph using k-nearest neighbors.

3.2 Models

We evaluate the models listed in Table 1 in all possible domains. As mentioned before all models working on *arbitrary* domains can be trained on both grid and cloud data by flattening the grid. Some architectures, such as Convolutional Neural Networks (CNNs) [17] and Residual Networks (ResNets) [11], are well established and may be used as a baseline comparison.

Fourier Neural Operator. Fourier Neural Operators (FNOs) [18] are a class of models designed to learn mappings between function spaces, particularly for solving partial differential equations. By operating in the Fourier domain, FNOs capture global dependencies efficiently, enabling fast and accurate predictions of complex physical systems. As mentioned in Section 2, we select the FNO architecture rather than any other Neural Operator model, as it has shown best performance before [15]. Lingsch et al. [22] extend FNOs to arbitrary structured data, which we denote as *GeoFNO*.

Graph Neural Networks (GNNs). This class of models includes Graph Convolutional Networks (GCN) [14], Graph Attention Networks (GAT) [32], FeaStNet [33], PointNet [28], PointGNN [29], and GraphPDE [13]. Leveraging

the intrinsic structure of graphs, these models employ local operations, such as neighborhood aggregation, to iteratively update node representations. Following the model selection from the original Dynabench benchmark [6], we include them in our experiments.

Point Transformer. The Point Transformer (PT) architecture was originally developed for point cloud processing [37]. It extends the Transformer framework to irregular 3D point sets, employing self-attention mechanisms combined with geometric priors to capture both local and global context. This enables effective learning of spatial relationships for tasks such as segmentation and classification. Point Transformer Version 3 (PTv3) [34] prioritizes efficiency over accuracy, allowing for significant scalability.

NeuralPDE and GrINd. The NeuralPDE [5] is a model that combines the CNN architecture and the method of lines to model arbitrarily complex PDEs. The model is limited to data sampled on a discrete spatial grid. To overcome this limitation, the GrINd model [4] that combines a Fourier interpolation layer with a NeuralPDE is used.

All models are implemented in PyTorch, utilizing PyTorch Lightning [7] to provide a simple training and testing interface for all models. Additionally, all graph models are implemented using PyTorch-Geometric [8]. The configuration of each experiment is done using Hydra [36] a framework for the configuration of complex applications in Python. This enables us to have an efficient training and evaluation pipeline independent of the model and dataset within Dynabench. The code is open-source and available online.

3.3 Evaluation Scheme

We evaluate all models across every combination of supported equations, resolutions, and domains. Specifically, each model is assessed at all four resolutions for all six equations within the grid domain, and, where applicable, on the cloud-structured data. Models for which the domain is not relevant are marked as having an arbitrary domain in Table 1. We closely follow the evaluation scheme that was used in [6].

For fairer comparison, we choose to keep the number of trainable parameters around 2 million for all models. An overview of the parameters that were chosen for each model can be found in the code repository[3]. Each model is typically trained for 10 epochs, using the entire training split of the dataset in each epoch, and is subsequently evaluated on the complete test split. The only exception is the GrINd model, where the training for 10 epochs was not feasible due to a high computational cost of the model and therefore reduced to 3 epochs. During training, the task is to learn the one-step temporal evolution (rollout $R = 1$) of the PDE given a certain initial condition. The input for the models is an H-step lookback of the system state (the previous H states). The additional lookback steps are concatenated along the feature dimension. We choose to include $H = 4$

[3] `https://professor-x.de/low_data_benchmark`

past states of the physical system. We train all models by minimizing the mean squared error (MSE):

$$\min_{\phi} \mathbb{E}\left[m_\phi\left(X_t||X_{t-1}||\dots||X_{t-H+1}\right) - X_{t+1}\right]^2 \tag{1}$$

where $X_{t+1}, X_t, X_{t-1}, \dots$ describe the states of the system at times $t+1, t, t-1, \dots$ and m_ϕ is the neural network with trainable parameter ϕ.

We evaluate the models on their performance of predicting the next $R = 16$ temporal steps for an initial condition using the MSE

$$\mathbb{E}\left[\hat{X}_{t+R} - X_{t+R}\right]^2, \tag{2}$$

the training time for a single training step, and the inference time during the evaluation. The prediction of the next steps is done in a closed-loop fashion, such that the output of the last step is part of the input to the next step.

$$\begin{aligned}
\hat{X}_{t+1} &= m_\phi\left(X_t||X_{t-1}||\dots||X_{t-H+1}\right) \\
\hat{X}_{t+2} &= m_\phi\left(\hat{X}_{t+1}||X_t||\dots||X_{t-H+2}\right) \\
\hat{X}_{t+3} &= m_\phi\left(\hat{X}_{t+2}||\hat{X}_{t+1}||\dots||X_{t-H+3}\right) \\
&\vdots \\
\hat{X}_{t+R} &= m_\phi\left(\hat{X}_{t+R-1}||\hat{X}_{t+R-2}||\dots||\hat{X}_{t-H+R}\right)
\end{aligned} \tag{3}$$

All models are trained with a learning rate of 1e−3 and L_2-regularization with weight 1e−6. The batch size is chosen to utilize the given hardware best: *low* and *medium* resolution jobs are generally trained with a batch size of 128 while *high* and *full* jobs with 64 and 32 respectively. We evaluate the model's best fit during training, based on the lowest MSE that we measured during validation four times per epoch. Moreover, we report the MSE for the persistence ($\hat{X}_{t+1} = \hat{X}_t$) and zero ($\hat{X}_{t+1} = \mathbf{0}$) baselines. We utilize two different computational clusters and therefore have two different types of GPUs that are used. Approximately half of the experiments are computed on NVIDIA L40 and the other half on NVIDIA A40 GPUs. We recognize that this makes the comparability of training and inference time among the models more complicated. However, many experiments within one resolution or model are consistent.

4 Results

We now evaluate the performance of all models using the inference loss for one and sixteen prediction steps, similar to [6], and compare the results with the persistence and zero baseline. Similar to the original DynaBench benchmark we show the results for the *high* resolution. In general, similar patterns can be observed for the 1 and 16 step rollout MSE for the *low* and *medium* resolutions (see the code repository[4] for details).

[4] `https://professor-x.de/low_data_benchmark`

Table 2: MSE after **1 prediction step** for the ***high*** resolution. The best performing grid model for each equation has been underlined. Additionally, the best non-grid model has been underwaved.

model	domain	A	B	GD	KS	RD	W
CNN	grid	1.63e−6	1.62e−4	2.54e−3	2.35e−4	1.30e−4	8.75e−4
ResNet	grid	4.23e−7	1.44e−4	2.06e−3	2.81e−4	6.02e−5	4.08e−4
FNO	grid	1.03e−5	2.21e−4	2.28e−3	1.20e−3	5.86e−5	1.31e−4
NeuralPDE	grid	5.26e−7	1.53e−4	2.79e−3	2.49e−4	1.22e−4	6.64e−4
GrINd	grid	9.36e−6	1.87e−2	6.82e−2	4.77e−1	2.01e−1	2.40e−3
PTv1	grid	3.98e−5	6.09e−4	4.70e−3	1.05e−3	2.11e−4	4.44e−4
PTv3	grid	1.49e−4	9.35e−4	6.22e−1	8.45e−3	3.77e−4	1.56e−3
GeoFNO	grid	3.42e−3	1.72e−3	7.15e−2	8.51e−2	8.43e−2	2.58e−4
PointNet	grid	1.00	1.00	9.99e−1	1.00	9.99e−1	1.00
PointGNN	grid	2.12e−4	4.80e−3	1.86e−1	1.00e−2	1.24e−4	2.56e−3
GraphPDE	grid	1.90e−5	4.65e−3	6.01e−2	1.27e−3	7.36e−5	1.72e−3
GAT	grid	1.52e−3	1.69e−3	2.57e−2	5.84e−2	1.24e−2	3.84e−3
GCN	grid	4.23e−3	1.40e−2	7.11e−2	4.63e−1	1.97e−1	1.20e−2
FeaStNet	grid	6.23e−5	6.75e−4	1.13e−2	2.30e−3	2.67e−4	9.04e−4
GrINd	cloud	7.18e−6	1.76e−2	6.94e−2	4.35e−1	1.83e−1	2.03e−3
PTv1	cloud	4.99e−4	1.13e−3	8.09e−3	4.42e−3	1.01e−4	5.58e−4
PTv3	cloud	4.38e−3	7.63e−2	1.69e−2	4.05e−3	1.98e−4	1.18e−3
GeoFNO	cloud	4.78e−4	1.94e−3	1.46e−2	1.24e−1	1.46e−1	1.15e−3
PointNet	cloud	1.00	1.00	9.98e−1	1.00	9.99e−1	1.00
PointGNN	cloud	2.03e−4	5.29e−3	2.03e−1	1.01e−2	1.51e−4	2.51e−3
GraphPDE	cloud	3.12e−5	1.68e−3	7.34e−2	3.69e−3	8.43e−5	1.51e−3
GAT	cloud	8.85e−3	5.02e−3	7.18e−2	1.19e−1	1.16e−2	1.33e−2
GCN	cloud	1.45e−2	2.95e−2	1.21e−1	5.00e−1	2.74e−1	3.82e−2
FeaStNet	cloud	1.26e−4	1.80e−3	2.41e−2	3.86e−3	3.56e−4	1.48e−3
persistence	baseline	8.19e−2	1.03e−2	3.11e−1	9.23e−2	1.14e−1	1.45e−1
zero	baseline	1.00	1.00	9.98e−1	1.00	9.99e−1	1.00

4.1 Performance Evaluation for *high* Resolution

In Table 2, the MSE after a single prediction step in the *high* resolution can be seen. The models are evaluated in the specified domain (grid or cloud) for the given equations abbreviated by the capital letters introduced in Section 3.1. We use color coding to mark the relative rank of each model, for easier comparison. The dark green color marks the best performing model, and the dark red the worst performing one. Note that the zero baseline consistently has an MSE of 1 due to the normalization of the dataset.

Models in the grid domain, such as CNN, ResNet, and FNO, outperform all other models in the cloud domain, achieving several orders of magnitude lower MSE. Models working directly on grid-structured data do not have to learn the spatial relationships between locations explicitly. We hypothesize that this

Table 3: MSE after **16 prediction steps** for the ***high*** resolution. The best performing grid model for each equation has been underlined. Additionally, the best non-grid model has been underwaved. Models with an MSE higher than 2 are marked as divergent (div).

model	domain	A	B	GD	KS	RD	W
CNN	grid	9.44e−5	2.80e−2	1.86e−1	5.71e−1	5.80e−2	7.21e−2
ResNet	grid	3.08e−5	1.74e−2	1.56e−1	5.63e−1	2.69e−2	3.79e−2
FNO	grid	1.50e−3	3.73e−2	9.42e−2	1.13	8.80e−2	4.74e−3
NeuralPDE	grid	2.70e−5	2.55e−2	2.02e−1	6.62e−1	4.99e−2	6.62e−2
GrINd	grid	2.75e−2	1.06e−1	5.38e−1	1.48	1.67e−1	1.51e−1
PTv1	grid	9.20e−1	4.07e−1	4.14e−1	1.34	6.53e−1	7.85e−1
PTv3	grid	div	1.73	div	1.94	6.19e−1	div
GeoFNO	grid	8.62e−1	1.99e−1	div	div	div	1.07e−2
PointNet	grid	1.00	1.00	9.99e−1	1.00	1.00	1.00
PointGNN	grid	1.25	7.28e−1	1.29	1.22	6.79e−1	1.18
GraphpPDE	grid	7.15e−1	3.08e−1	8.79e−1	1.56	8.95e−2	9.28e−1
GAT	grid	9.53e−1	div	div	div	div	div
GCN	grid	div	1.54	div	div	1.19	div
FeaStNet	grid	div	1.41e−1	8.07e−1	1.60	1.38e−1	2.37e−1
GrINd	cloud	6.58e−3	1.18e−1	3.02e−1	1.35	1.92e−1	1.34e−1
PTv1	cloud	8.13e−1	3.28e−1	4.62e−1	1.27	2.72e−1	6.38e−1
PTv3	cloud	div	div	5.11e−1	1.25	1.75e−1	8.10e−1
GeoFNO	cloud	6.44e−3	2.63e−1	1.93e−1	div	div	5.36e−2
PointNet	cloud	1.00	1.00	9.98e−1	1.00	1.00	1.00
PointGNN	cloud	1.21	3.21e−1	1.15	1.12	2.73e−1	1.29
GraphPDE	cloud	5.03e−1	1.91e−1	9.96e−1	1.79	1.10e−1	9.08e−1
GAT	cloud	1.16	6.20e−1	1.94	1.80	1.19	div
GCN	cloud	div	9.71e−1	1.67	1.27	1.02	1.35
FeaStNet	cloud	1.07	2.50e−1	5.93e−1	1.63	2.08e−1	5.42e−1
persistence	baseline	2.39	3.47e−1	1.84	1.85	1.58	2.33
zero	baseline	1.00	1.00	9.98e−1	1.00	1.00	1.00

allows the grid models to allocate more learning capacity toward forecasting the system's dynamics, thereby making the task comparatively easier. Further, the Point Transformer version 1 (PTv1) outperforms other models in the cloud domain for three equations (Burgers, Gas-Dynamics, and Wave) and has only a marginally higher MSE than the best-performing models for the other equations. In general, the complexity of the equation directly influences the performance of the models (c.f. the Advection equation, a single linear PDE and the Gas-Dynamics equation, a system of four non-linear PDEs).

In Table 3, the MSE after sixteen prediction steps in the *high* resolution can be seen. Models are marked as divergent (div) if the MSE is above the threshold of 2. Again, models that require the grid domain perform far better than all other models, indicated by the green color. For the cloud-structured

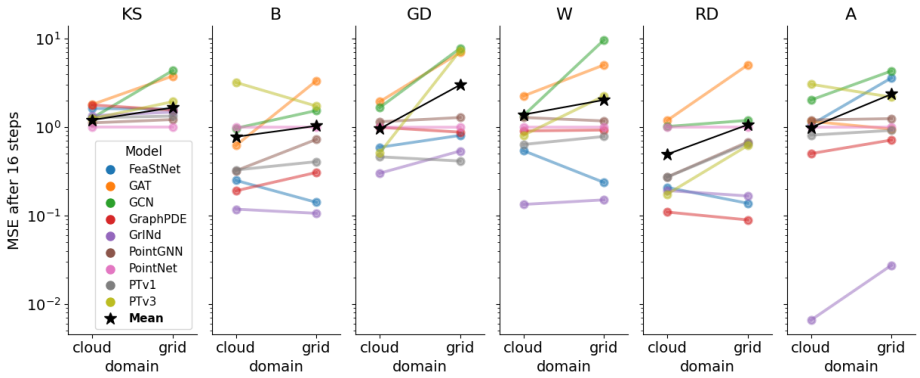

Fig. 2: Comparison of the performance of arbitrary domain models on grid and cloud structured data as measured by the error after 16 rollout steps on the *high* resolution. The results show that these types of models perform better on cloud structured data on average. GeoFNO has been excluded, as it diverges for 3 out of the 6 equations.

data, the GeoFNO performs very well in the Advection, Gas-Dynamics, and Wave equations, but diverges for the Kuramoto-Sivashinsky (KS) and Reaction-Diffusion (RD) equations. Our experiments show that the KS equation proves to be particularly challenging for the arbitrary domain models, as none of them achieves a lower MSE than the zero baseline. Interestingly, several models (i.e. GCN, GrINd) that were tested both in the cloud and the grid domain show better performance in the cloud domain (Figure 2).

In Figure 3, the MSE for the temporal evolution for each model in the grid and cloud domains can be seen. It shows the MSE for the Advection equation, where the MSE slowly increases with the number of steps for grid-based models. Most arbitrary domain models converge towards 1 with increasing number of steps, with the exception of GrINd and GeoFNO.

4.2 Influence of Resolution

We summarize the results for different resolutions of the data in Table 4. The mean rank over all resolutions gives a global rank over all parameters for each model. The highest ranks are occupied by ResNet and other grid-based models. Moreover, the GrINd, PTv1, and GraphPDE models show promising results in both the grid and cloud domains. In Figure 4, the median MSE for the models is shown. The median MSE is computed over all equations and domains for a single resolution. The left plot generally shows that higher resolution leads to a lower one-step MSE and therefore better model performance. However, for the 16-step median MSE, this behavior cannot be observed.

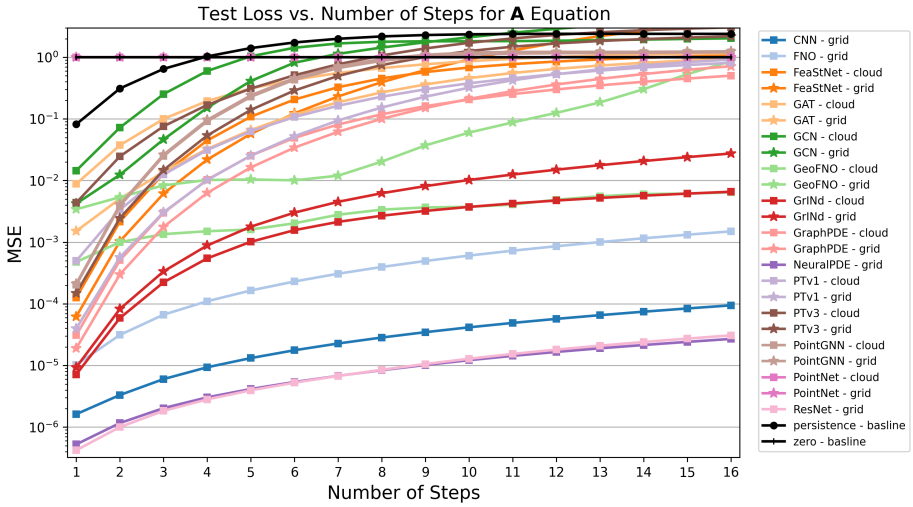

Fig. 3: Rollout MSE for the Advection (**A**) equation in the *high* resolution. Models are marked with a color, while the domain can be differentiated with the symbol (grid: ■ and cloud: ⋆).

5 Discussion

In the previous section, we presented the results of our benchmark experiments on the DynaBench dataset, analyzing the performance of various models across different resolutions, domains, and rollout lengths. Several key insights emerge from our evaluation.

First, we observe a strong influence of resolution on model performance for short-term predictions. For one-step prediction tasks, all models—regardless of architecture or domain—benefit from increased resolution, achieving lower mean squared error (MSE). However, this trend does not extend to longer rollouts. For sixteen-step predictions, cloud-based models exhibit significantly less stable behavior, often converging toward the zero baseline or diverging entirely. This suggests that unstructured data poses greater challenges for capturing long-term dynamics, potentially due to limitations in how these models represent spatial relationships. Grid-based models, by contrast, show decreasing MSE up to the *high* resolution. Interestingly, performance slightly deteriorates at the *full* resolution, implying that the chosen model capacity is not sufficient to learn the long term behavior of the systems. We hypothesize that selecting larger model architectures may mitigate this problem.

Regarding model architecture, grid-based models consistently outperform their cloud-based counterparts across all resolutions and equations. This advantage likely stems from the structured nature of the input data, which removes the need for these models to learn the spatial relationship between observation

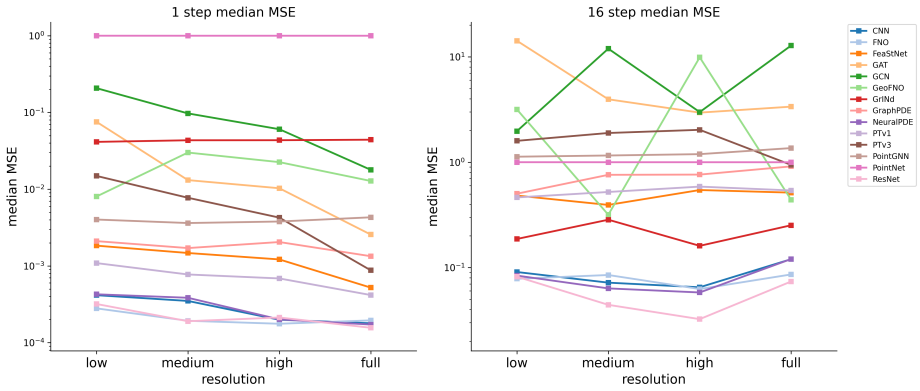

Fig. 4: Median MSE for all four resolutions for each model for single-step (left) and sixteen-step prediction (right). The Median is computed over all equations and domains within one resolution.

points. This effect is particularly evident in the case of the GrINd model, which interpolates irregularly spaced observations onto a regular grid to leverage a grid-based backbone. As a result, GrINd outperforms all other models operating on cloud-structured data and even rivals state-of-the-art grid-native models.

Interestingly, the structure of the input data—whether grid or cloud—has only a limited effect on one-step prediction performance. Models tested in both domains achieve comparable accuracy for short rollouts. However, for longer rollouts, cloud-structured input leads to marginally better performance for some arbitrary-domain models. This counterintuitive result suggests that certain architectures designed for unstructured data may be better able to exploit scattered information, such as local spatial gradients, to infer more stable long-term dynamics. While this effect is modest, it highlights the potential for future models to more effectively leverage the flexibility of cloud-based representations.

Finally, in terms of rollout stability, only grid-based models consistently produce stable long-term predictions. This includes GrINd, which, although operating on cloud data, ultimately applies a grid-based prediction model. Arbitrary-domain models generally fail to maintain predictive stability beyond a few steps, with many diverging or reverting to the baseline. This underscores the ongoing difficulty of learning reliable temporal evolution in unstructured domains with sparse observations, especially when training does not explicitly incorporate multi-step objectives.

Taken together, these findings emphasize the strengths of structured data representations and grid-based architectures for modeling spatiotemporal dynamics, while also pointing to opportunities for innovation in learning from irregular or unstructured data.

Table 4: This table displays the average rank (value between 0 and 25) of each model across all equations and resolutions for **sixteen prediction steps**. Additionally, the models are sorted by the mean rank across *low*, *medium*, and *high* resolutions.

model	domain	low	medium	high	mean	full
ResNet	grid	2.2	0.8	0.7	1.2	0.7
NeuralPDE	grid	4.8	1.7	2.0	2.8	1.7
CNN	grid	5.2	2.3	2.3	3.3	2.2
FNO	grid	4.5	3.2	2.7	3.4	2.3
GrINd	cloud	7.0	7.3	7.2	7.2	-
GrINd	grid	6.0	9.0	7.8	7.6	5.2
GraphPDE	grid	8.5	11.2	10.7	10.1	9.3
GraphPDE	cloud	8.7	11.3	10.5	10.2	-
FeaStNet	grid	9.5	10.8	11.8	10.7	7.3
PTv1	cloud	11.3	11.2	10.2	10.9	-
GeoFNO	grid	9.7	9.3	15.5	11.5	7.3
PTv1	grid	12.7	11.2	11.5	11.8	7.2
FeaStNet	cloud	12.2	13.2	11.8	12.4	-
GeoFNO	cloud	15.8	12.0	11.5	13.1	-
zero	baseline	13.2	12.8	13.8	13.3	8.5
PointGNN	cloud	12.5	13.8	14.0	13.4	-
PointNet	cloud	12.8	13.0	14.5	13.4	-
PointNet	grid	13.2	13.5	14.7	13.8	8.5
PTv3	cloud	14.8	14.5	14.0	14.4	-
PointGNN	grid	13.5	15.0	15.8	14.8	11.2
GCN	cloud	17.5	22.0	17.7	19.1	-
persistence	baseline	19.8	19.7	20.2	19.9	11.5
GAT	cloud	22.2	21.0	19.2	20.8	-
PTv3	grid	21.8	20.8	20.3	21.0	9.0
GAT	grid	23.2	20.8	21.3	21.8	13.2
GCN	grid	22.2	23.3	23.3	22.9	14.0

6 Conclusions

In this work, we have evaluated fourteen models on the entire DynaBench benchmark dataset. Our benchmark focuses on learning from low-resolution, sparse, and scattered data. The models were chosen based on their impact on current research developments and represent the major research fields, such as GNNs, Transformers, and Neural Operators. Our results show that grid-based models consistently outperform cloud-based ones in a low-data regime, particularly for long-term predictions, due to their ability to leverage structured spatial information. However, certain models designed for unstructured data, such as GrINd and Point Transformer, show promising performance, suggesting that with appropriate architectural design, cloud-based approaches can approach the effectiveness of grid-based models. In line with findings from weather and climate

modeling [1,25] our results highlight that achieving stable long-term rollouts is a difficult task, especially for sparse and scattered, low-resolution data.

Our study also reveals interesting research opportunities for improving the robustness and generalization of models operating on unstructured data. An interesting observation is that models that were originally developed for the grid domain and adapted to the cloud domain, such as GrINd and GeoFNO, perform well on a variety of different physical systems. Future work could explore more hybrid architectures that combine the spatial inductive biases of grid-based models with the flexibility of point-based methods, enabling more stable long-term predictions in arbitrary domains. Additionally, incorporating multi-step objectives during training, or leveraging physics-informed constraints, may enhance rollout stability and accuracy, particularly in data-scarce scenarios. Finally, scaling model capacity in conjunction with resolution, and developing adaptive methods that dynamically adjust to data structure and complexity, offer promising directions for further improving performance across a wide range of spatiotemporal forecasting tasks.

Disclosure of Interests. The project underlying this publication was funded by the German Federal Ministry of Education and Research under the grant number 16DKWN0099B (MAGNET4Cardiac7T). The responsibility for the content of this publication lies with the authors.

References

1. Bonev, B., Kurth, T., Hundt, C., Pathak, J., Baust, M., Kashinath, K., Anandkumar, A.: Spherical Fourier neural operators: Learning stable dynamics on the sphere. In: Krause, A., Brunskill, E., Cho, K., Engelhardt, B., Sabato, S., Scarlett, J. (eds.) Proceedings of the 40th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 202, pp. 2806–2823. PMLR (23–29 Jul 2023)
2. Bonnet, F., Mazari, J.A., Munzer, T., Yser, P., Gallinari, P.: An extensible benchmarking graph-mesh dataset for studying steady-state incompressible navier-stokes equations. arXiv preprint arXiv:2206.14709 (2022)
3. Della Mea, G.B., Ovalle, C., Laiarinandrasana, L., Decencière, E., Dokládal, P.: Physics informed self-supervised segmentation of elastic composite materials. Computer Methods in Applied Mechanics and Engineering **432**, 117355 (2024)
4. Dulny, A., Heinisch, P., Hotho, A., Krause, A.: GrINd: Grid Interpolation Network for Scattered Observations. In: Joint European Conference on Machine Learning and Knowledge Discovery in Databases. pp. 177–193. Springer (2024)
5. Dulny, A., Hotho, A., Krause, A.: NeuralPDE: Modelling dynamical systems from data. In: German Conference on Artificial Intelligence (Künstliche Intelligenz). pp. 75–89. Springer (2022)
6. Dulny, A., Hotho, A., Krause, A.: Dynabench: A benchmark dataset for learning dynamical systems from low-resolution data. In: Joint European Conference on Machine Learning and Knowledge Discovery in Databases. pp. 438–455. Springer (2023)
7. Falcon, W., The PyTorch Lightning team: PyTorch Lightning (Mar 2019)

8. Fey, M., Lenssen, J.E.: Fast Graph Representation Learning with PyTorch Geometric (May 2019)
9. Hanna, J.M., Aguado, J.V., Comas-Cardona, S., Askri, R., Borzacchiello, D.: Residual-based adaptivity for two-phase flow simulation in porous media using physics-informed neural networks. Computer Methods in Applied Mechanics and Engineering **396**, 115100 (2022)
10. Hasan, M., Ender Eger, Z., Senthilnathan, A., Acar, P.: Microstructure-sensitive deformation modeling and materials design with physics-informed neural networks. AIAA Journal **62**(5), 1864–1874 (2024)
11. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) pp. 770–778 (2015)
12. Huang, Z., Schneider, T., Li, M., Jiang, C., Zorin, D., Panozzo, D.: A large-scale benchmark for the incompressible navier-stokes equations. arXiv preprint arXiv:2112.05309 (2021)
13. Iakovlev, V., Heinonen, M., Lähdesmäki, H.: Learning continuous-time PDEs from sparse data with graph neural networks
14. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks
15. Kovachki, N., Li, Z., Liu, B., Azizzadenesheli, K., Bhattacharya, K., Stuart, A., Anandkumar, A.: Neural operator: Learning maps between function spaces with applications to pdes. Journal of Machine Learning Research **24**(89), 1–97 (2023)
16. Lam, R., Sanchez-Gonzalez, A., Willson, M., Wirnsberger, P., Fortunato, M., Alet, F., Ravuri, S., Ewalds, T., Eaton-Rosen, Z., Hu, W., et al.: Learning skillful medium-range global weather forecasting. Science **382**(6677), 1416–1421 (2023)
17. Lecun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proceedings of the IEEE **86**(11), 2278–2324 (1998-11)
18. Li, Z., Kovachki, N., Azizzadenesheli, K., Liu, B., Bhattacharya, K., Stuart, A., Anandkumar, A.: Fourier neural operator for parametric partial differential equations. arXiv preprint arXiv:2010.08895 (2020)
19. Li, Z., Kovachki, N., Azizzadenesheli, K., Liu, B., Bhattacharya, K., Stuart, A., Anandkumar, A.: Neural operator: Graph kernel network for partial differential equations (2020)
20. Li, Z., Kovachki, N., Azizzadenesheli, K., Liu, B., Stuart, A., Bhattacharya, K., Anandkumar, A.: Multipole graph neural operator for parametric partial differential equations. In: Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., Lin, H. (eds.) Advances in Neural Information Processing Systems. vol. 33, pp. 6755–6766. Curran Associates, Inc. (2020)
21. Lienen, M., Lüdke, D., Hansen-Palmus, J., Günnemann, S.: From zero to turbulence: Generative modeling for 3d flow simulation. In: The Twelfth International Conference on Learning Representations (2024)
22. Lingsch, L.E., Michelis, M.Y., De Bezenac, E., M. Perera, S., Katzschmann, R.K., Mishra, S.: Beyond Regular Grids: Fourier-Based Neural Operators on Arbitrary Domains. In: Salakhutdinov, R., Kolter, Z., Heller, K., Weller, A., Oliver, N., Scarlett, J., Berkenkamp, F. (eds.) Proceedings of the 41st International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 235, pp. 30610–30629. PMLR (2024)
23. Lu, L., Jin, P., Pang, G., Zhang, Z., Karniadakis, G.E.: Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators. Nature Machine Intelligence **3**(3), 218–229 (2021-03)

24. Luo, Y., Chen, Y., Zhang, Z.: Cfdbench: A large-scale benchmark for machine learning methods in fluid dynamics. arXiv preprint arXiv:2310.05963 (2023)
25. McCabe, M., Harrington, P., Subramanian, S., Brown, J.: Towards stability of autoregressive neural operators (2023)
26. Nguyen, T., Brandstetter, J., Kapoor, A., Gupta, J.K., Grover, A.: ClimaX: A foundation model for weather and climate. In: Proceedings of the 40th International Conference on Machine Learning. pp. 25904–25938. PMLR (2023)
27. Pathak, J., Subramanian, S., Harrington, P., Raja, S., Chattopadhyay, A., Mardani, M., Kurth, T., Hall, D., Li, Z., Azizzadenesheli, K., et al.: Fourcastnet: A global data-driven high-resolution weather model using adaptive fourier neural operators. arXiv preprint arXiv:2202.11214 (2022)
28. Qi, C.R., Su, H., Mo, K., Guibas, L.J.: Pointnet: Deep learning on point sets for 3d classification and segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 652–660 (2017)
29. Shi, W., Rajkumar, R.: Point-gnn: Graph neural network for 3d object detection in a point cloud. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 1711–1719 (2020)
30. Takamoto, M., Praditia, T., Leiteritz, R., MacKinlay, D., Alesiani, F., Pflüger, D., Niepert, M.: PDEBench: An extensive benchmark for scientific machine learning. In: Koyejo, S., Mohamed, S., Agarwal, A., Belgrave, D., Cho, K., Oh, A. (eds.) Advances in Neural Information Processing Systems. vol. 35, pp. 1596–1611. Curran Associates, Inc. (2022)
31. Toshev, A., Galletti, G., Fritz, F., Adami, S., Adams, N.: LagrangeBench: A lagrangian fluid mechanics benchmarking suite. In: Oh, A., Naumann, T., Globerson, A., Saenko, K., Hardt, M., Levine, S. (eds.) Advances in Neural Information Processing Systems. vol. 36, pp. 64857–64884. Curran Associates, Inc. (2023)
32. Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., Bengio, Y.: Graph attention networks
33. Verma, N., Boyer, E., Verbeek, J.: Feastnet: Feature-steered graph convolutions for 3d shape analysis. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 2598–2606 (2018)
34. Wu, X., Jiang, L., Wang, P.S., Liu, Z., Liu, X., Qiao, Y., Ouyang, W., He, T., Zhao, H.: Point transformer v3: Simpler faster stronger. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 4840–4851 (2024)
35. Xing, L., Wu, H., Ma, Y., Wang, J., Long, M.: Helmfluid: learning helmholtz dynamics for interpretable fluid prediction. In: Proceedings of the 41st International Conference on Machine Learning. pp. 54673–54697 (2024)
36. Yadan, O.: Hydra - a framework for elegantly configuring complex applications. Github (2019)
37. Zhao, H., Jiang, L., Jia, J., Torr, P.H., Koltun, V.: Point transformer. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 16259–16268 (2021)