Continuous PDE Dynamics Forecasting with Implicit Neural Representations

Yuan Yin[*1] **Matthieu Kirchmeyer**[*1,2] **Jean-Yves Franceschi**[*2]
Alain Rakotomamonjy[2] **Patrick Gallinari**[1,2]
[1]Sorbonne Université, CNRS, ISIR, F-75005 Paris, France [2]Criteo AI Lab, Paris

Abstract

Effective data-driven PDE forecasting methods often rely on fixed spatial and / or temporal discretizations. This raises limitations in real-world applications like weather prediction where flexible extrapolation at arbitrary spatiotemporal locations is required. We address this problem by introducing a new data-driven approach, DINo, that models a PDE's flow with continuous-time dynamics of spatially continuous functions. This is achieved by embedding spatial observations independently of their discretization via Implicit Neural Representations in a small latent space temporally driven by a learned ODE. This separate and flexible treatment of time and space makes DINo the first data-driven model to combine the following advantages. It extrapolates at arbitrary spatial and temporal locations; it can learn from sparse irregular grids or manifolds; at test time, it generalizes to new grids or resolutions. DINo outperforms alternative neural PDE forecasters in a variety of challenging generalization scenarios on representative PDE systems.

1 Introduction

Modeling the dynamics and predicting the temporal evolution of physical phenomena is paramount in many fields, e.g. climate modeling, biology, fluid mechanics and energy [47]. Classical solutions rely on a well-established physical paradigm: the evolution is described by differential equations derived from physical first principles, and then solved using numerical analysis tools, e.g. finite elements, finite volumes or spectral methods [33]. The availability of large amounts of data from observations or simulations has motivated data-driven approaches to this problem [5], leading to a rapid development of the field with deep learning. The main motivations for this research track include developing surrogate or reduced order models that can approximate high-fidelity full order models at reduced computational costs [23], complementing classical solvers, e.g. to account for additional components of the dynamics [49], or improving low fidelity models [10].

Most of these attempts rely on workhorses of deep learning like CNNs [1] or GNNs [26, 37, 4]. They all require prior space discretization either on regular or irregular grids, such that they only capture the dynamics on the train grid and cannot generalize outside it. Neural operators, a recent trend, learn mappings between function spaces [28, 31] and thus alleviate some limitations of prior discretization approaches. Yet, they still rely on fixed grid discretization for training and inference: e.g., regular grids for [28] or a free-form but predetermined grid for [31]. Hence, the number and / or location of the sensors has to be fixed across train and test which is restrictive in many situations [38]. Mesh-agnostic approaches for solving canonical PDEs (Partial Differential Equations) are another trend [39, 40]. In contrast to physics-agnostic grid-based approaches, they aim at solving a known PDE as usual solvers do, and cannot cope with unknown dynamics. This idea was concurrently developed for computer graphics, e.g. for learning 3D shapes [41, 32, 44] and coined as Implicit

*Equal contribution.

NeurIPS 2022 AI for Science Workshop.

Table 1: Comparison of data-driven approaches to spatiotemporal PDE forecasting.

	Model	Ref.	1. PDE-agnostic prediction on new initial conditions	2. Train / test space grid independence	3. Evaluation at unobserved spatial locations	4. Free-form spatial grid and topology	5. Time continuous	6. Time extrapolation
Discrete	NODE	[9]	✓	✗	✗	✗	✓	✓
	MP-PDE	[4]	✓	✗	✗	✓	✗	✓
Operator	MNO	[27]	✓	✓	✗	✗	✗	✓
	DeepONet	[31]	✓	✗	✓	✓	✓	✗
INRs	PINNs	[39]	✗	✓	✓	✓	✓	✗
	DINo	Ours	✓	✓	✓	✓	✓	✓

Neural Representations (INRs). When used as solvers, these methods can only tackle a single initial value problem and are not designed for long-term forecasting outside the training horizon.

Due to these limitations, these approaches cannot handle many practical applications such as: different geometries, e.g. PDEs lying on a Euclidean plane or an Earth-like sphere; variable sampling, e.g. irregular observation grids that evolve as in adaptive meshing [3]; scarce training data e.g. with available observations at few spatiotemporal locations; multi-scale phenomena, e.g. in climate modeling, where integrating intertwined subgrid scales a.k.a. the closure problem is ubiquitous [51]. This motivates new machine learning models which improve existing approaches on these aspects.

In our work, we aim at forecasting PDE-based spatiotemporal physical processes with a versatile model tackling the aforementioned limitations. We adopt an agnostic approach, i.e. not assuming any prior knowledge on the physics. We introduce DINo (Dynamics-aware Implicit Neural representations), a model operating continuously in space and time, with the following contributions.

Continuous flow learning. DINo aims at learning the PDE's flow to forecast its solutions, in a continuous manner so that it can be trained on any spatial and temporal discretization and applied to another. To this end, DINo embeds spatial observations into a small latent space via INRs; then it models continuous-time evolution by a learned latent Ordinary Differential Equation (ODE).
Space-time separation. To efficiently encode different sequences, we propose a novel INR parameterization, amplitude modulation, implementing a space-time separation of variables. This simplifies the learned dynamics, reduces the number of parameters and greatly improves performance.
Spatiotemporal versatility. DINo combines the benefits of prior models, cf. Table 1. It tackles new sequences via its amplitude modulation. Sequential modeling with an ODE makes it extrapolate to unseen times within or beyond the train horizon. Thanks to INRs' spatial flexibility, it generalizes to new grids or resolutions, predicts at arbitrary positions and handles sparse irregular grids or manifolds.
Empirical validation. We demonstrate DINo's versatility and state-of-the-art performance vs prior neural PDE forecasters, representative of grid-based, operator and INR-based methods, via thorough experiments on challenging multi-dimensional PDEs in various spatiotemporal generalization settings.

2 Problem Description

Problem setting. We aim at modeling, via a data-driven approach, the temporal evolution of a continuous fully-observed spatiotemporal phenomenon. It is described by trajectories $v\colon \mathbb{R} \to \mathcal{V}$ in a set $\mathit{\Gamma}$; we use $v_t \triangleq v(t) \in \mathcal{V}$. Trajectories share the same dynamics but differ by their initial condition $v_0 \in \mathcal{V}$. $\mathbb{R}$ is the temporal domain and $\mathcal{V}$ is the functional space of the form $\Omega \to \mathbb{R}^n$, where $\Omega \subset \mathbb{R}^p$ is a compact domain of spatial coordinates and n the number of observed values. In other words, v_t is a spatial function of $x \in \Omega$, with vectorial output $v_t(x) \in \mathbb{R}^n$; cf. examples of Section 4.1. To this end, we consider the setting illustrated in Figure 1. We observe a finite training set of trajectories $\mathcal{D}$, with a free-formed spatial observation grid $\mathcal{X}_{\text{tr}} \subset \Omega$ and on discrete times $t \in \mathcal{T} \subset [0, T]$. At test time, we are only given a new initial condition v_0, with observed values $v_0|_{\mathcal{X}_{\text{ts}}}$ restricted to a new observation grid $\mathcal{X}_{\text{ts}}$, potentially different from $\mathcal{X}_{\text{tr}}$. Inference is performed on both train and test trajectories given only the initial condition, on a new free-formed grid $\mathcal{X}' \subset \Omega$ and times $t \in \mathcal{T}' \subset [0, T']$. Inference grid $\mathcal{X}'$ comprises observed positions (respectively $\mathcal{X}_{\text{tr}}$ and $\mathcal{X}_{\text{ts}}$ for train and test trajectories) and unobserved positions corresponding to spatial extrapolation. Note that the inference temporal horizon is larger than the train one: $T < T'$. For simplicity, *In-s* refers to data in $\mathcal{X}'$ on the observation grid ($\mathcal{X}_{\text{tr}}$ for *train* / $\mathcal{X}_{\text{ts}}$ for *test*), *Out-s* to data in $\mathcal{X}'$ outside the observation grid; *In-t* refers to times within the train horizon $\mathcal{T} \subset [0, T]$, and *Out-t* to times in $\mathcal{T}' \backslash \mathcal{T} \subset (T, T']$, beyond T, up to inference horizon T'.

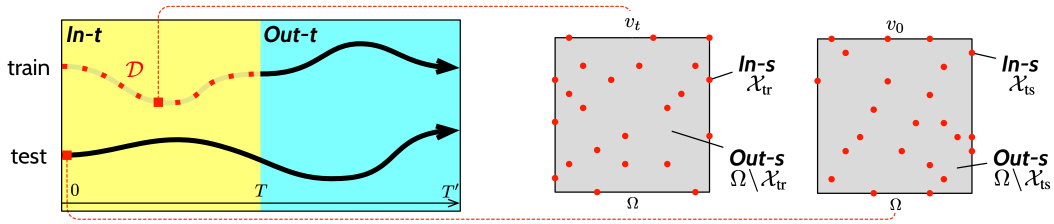


Figure 1: (Left) We represent time contexts. The *train* trajectory consists of training snapshots (■), observed in a train interval $[0, T]$ denoted *In-t*. The line (—) in continuation is a forecasting of this trajectory beyond *In-t*, in $(T, T']$ denoted *Out-t*. The line below (—, *test*) is a forecasting from a new initial condition v_0 (■) on *In-t* and *Out-t*. (Middle and right) We illustrate spatial contexts. (Middle) Dots (●) correspond to the train observation grid $\mathcal{X}_{\text{tr}}$, denoted *In-s*. *Out-s* denotes the complementary domain $\Omega \setminus \mathcal{X}_{\text{tr}}$. (Right) New test observation grid $\mathcal{X}_{\text{ts}}$, used as an initial point for forecasting (left).

Evaluation scenarios. The desired properties in Section 1 call for space-time continuous forecasters. We select six criteria that should be met; cf. column titles of Table 1. First, the model should generalize to new initial conditions v_0, i.e. to *test* trajectories (col. 1). Second, it should extrapolate beyond the train conditions: in space, on a test observation grid that differs from the train one, i.e. $\mathcal{X}' = \mathcal{X}_{\text{ts}} \neq \mathcal{X}_{\text{tr}}$ (*In-s*) (col. 2), and outside the observed train and test grid, i.e. on $\mathcal{X}' \setminus \mathcal{X}_{\text{ts}}, \mathcal{X}' \setminus \mathcal{X}_{\text{tr}}$ (*Out-s*, col. 3); in time, between train snapshots (col. 5) and beyond the train horizon T (*Out-t*, col. 6). Finally, it should adapt to free-form spatial domains, i.e. to various geometries (e.g. manifolds) or irregular grids (col. 4).

Objective. To satisfy these requirements, we learn the flow Φ of the physical system:

$$\Phi\colon (\mathcal{V} \times \mathbb{R}) \to \mathcal{V}, \qquad (v_t, \tau) \mapsto \Phi_\tau(v_t) = v_{t+\tau} \quad \forall v \in \Gamma, t \in \mathbb{R}. \tag{1}$$

Learning the flow is a common strategy in sequential models to better generalize beyond the train time horizon. Yet, so far, it has always been learned with discretized models, which poses generalization issues violating our requirements. We describe these issues in our related work in Section A.

3 Model

We present DINo, the first space / time-continuous model that tackles all prediction tasks of Section 2, without the limitations of prior approaches (Section A). We specify DINo's inference procedure (Section 3.1, Figure 2 left), then introduce its components (Section 3.2) and how they are trained (Section 3.3, Figure 2 right). Finally, we detail our implementation based on amplitude modulation, a novel INR parameterization for spatiotemporal data performing separation of variables (Section 3.4).

3.1 Inference Model

As explained in Section 2, we aim at estimating the flow Φ in Eq. (1), so that our model can be trained on an observed grid $\mathcal{X}_{\text{tr}}$ and perform inference given a new one $\mathcal{X}_{\text{ts}}$, both possibly irregular. To this end, we leverage a space- and time-continuous formulation, independent of a given data discretization. At inference, DINo starts from an initial condition $v_0 \in \mathcal{V}$ and uses a flow to forecast its dynamics. DINo first embeds spatial observations from v_0 into a latent vector α_0 of small dimension via an encoder of spatial functions $E_\varphi\colon \mathcal{V} \to \mathbb{R}^{d_\alpha}$ (ENC). Then, it unrolls a latent time-continuous dynamics model $f_\psi\colon \mathbb{R}^{d_\alpha} \to \mathbb{R}^{d_\alpha}$ given this initial condition (DYN). Finally, it decodes latent vectors via a decoder $D_\phi\colon \mathbb{R}^{d_\alpha} \to \mathcal{V}$ into a function of space (DEC). At any time t, D_ϕ takes as input α_t and outputs a function $\tilde{v}_t\colon \Omega \to \mathbb{R}^n$. This results in the following model, illustrated in Figure 2 (left):

$$\text{(ENC)}\ \alpha_0 = E_\varphi(v_0), \qquad \text{(DYN)}\ \frac{\mathrm{d}\alpha_t}{\mathrm{d}t} = f_\psi(\alpha_t), \qquad \text{(DEC)}\ \forall t, \tilde{v}_t = D_\phi(\alpha_t). \tag{2}$$

3.2 Components

We now further detail each component involved at inference from Eq. (2).

Encoder: $\alpha_t = E_\varphi(v_t)$. The encoder computes a latent vector α_t given observation v_t at any time t. It is used in two different contexts, respectively for train and test. At train time, given an observed trajectory $v_\mathcal{T} = \{v_t\}_{t \in \mathcal{T}}$, it will encode any v_t into α_t (see Section 3.3). At inference time, only v_0

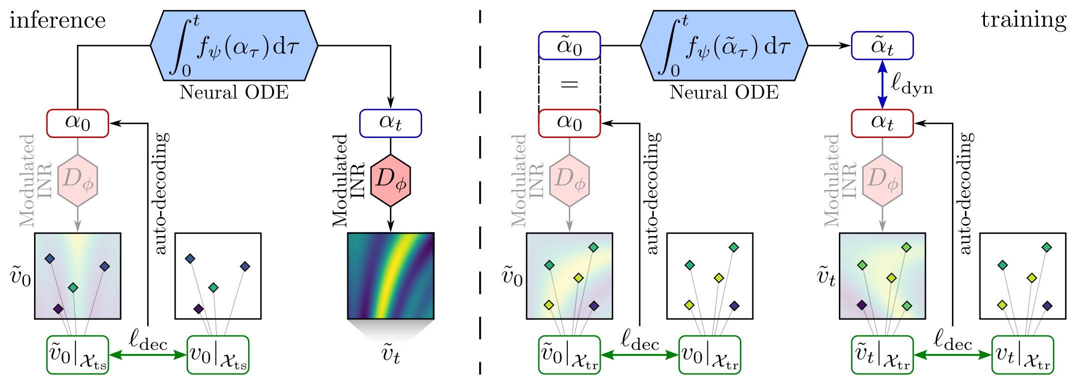


Figure 2: Proposed DINo model. Inference (left): given a new initial condition observed on a grid $\mathcal{X}_{\text{ts}}$, $v_0|_{\mathcal{X}_{\text{ts}}}$, forecasting amounts at decoding α_t to $\tilde{v}_t$, by unrolling α_0 with a time-continuous ODE dynamics model f_ψ. Train (right): given an observation grid $\mathcal{X}_{\text{tr}}$ and a space-continuous decoder D_ϕ, α_t is learned by auto-decoding s.t. $D_\phi(\alpha_t)|_{\mathcal{X}_{\text{tr}}} = v_t|_{\mathcal{X}_{\text{tr}}}$. Its evolution is then modelled with f_ψ.

is available, and then only α_0 is computed to be used as initial value for the dynamics. Given the decoder D_ϕ, α_t is a solution to the inverse problem $D_\phi(\alpha_t) = v_t$. We solve this inverse problem with auto-decoding [34]. Denoting $\ell_{\text{dec}}(\phi, \alpha_t; v_t) = \|D_\phi(\alpha_t) - v_t\|_2^2$ the decoding loss where $\|\cdot\|_2$ is the euclidean norm of a function and K the number of update steps, auto-decoding defines E_φ as:

$$E_\varphi(v_t) = \alpha_t^K, \quad \text{where} \quad \forall k > 1, \alpha_t^k = \alpha_t^{k-1} - \eta \nabla_{\alpha_t} \ell_{\text{dec}}\Big(\phi, \alpha_t^{k-1}; v_t\Big) \quad \text{and} \quad \varphi = \phi. \tag{3}$$

In practice, we observe a discretization $(\mathcal{X}_{\text{tr}}, \mathcal{X}_{\text{ts}})$ and accordingly approximate the norm in ℓ_{dec} as in Eq. (6). Compared to auto-encoding, auto-decoding underfits less [20] and is more flexible: without requiring specialized encoders, it handles free-formed observation grids (irregular or on a manifold) with an appropriate decoder.

Decoder: $\tilde{v}_t = D_\phi(\alpha_t)$**.** We define a flexible decoder using an INR. An INR I_θ is a space-continuous function parameterized by $\theta \in \mathbb{R}^{d_\theta}$ defined on domain Ω as illustrated in Figure 3. INRs approximate functions independently of the observation grid, e.g. it handles irregular grids and changing observation positions unlike FNO and DeepONet. Thus, it constitutes a flexible alternative to operators suitable to auto-decoding. We condition the INR's parameters θ on α_t via a hypernetwork [17] $h_\phi \colon \mathbb{R}^{d_\alpha} \to \mathbb{R}^{d_\theta}$. It generates high-dimensional parameters $\theta_t \in \mathbb{R}^{d_\theta}$ of the INR given the low-dimensional latent vector $\alpha_t \in \mathbb{R}^{d_\alpha}$, i.e. $\theta_t = h_\phi(\alpha_t)$. In summary, the decoder D_ϕ, parameterized as h by ϕ, is defined as follows (cf. details in Section 3.4):

$$\forall x \in \Omega, \quad \tilde{v}_t(x) = D_\phi(\alpha_t)(x) \triangleq I_{h_\phi(\alpha_t)}(x). \tag{4}$$

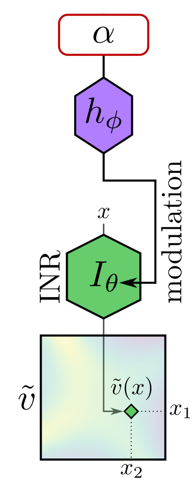


Fig. 3: Decoder D_ϕ.

Dynamics model: $\frac{\mathrm{d}\alpha_t}{\mathrm{d}t} = f_\psi(\alpha_t)$**.** The dynamics model $f_\psi \colon \mathbb{R}^{d_\alpha} \to \mathbb{R}^{d_\alpha}$ defines a flow via a latent ODE. An initial condition can be defined for all t by encoding with E_φ the input function v_t. In practice, to unroll the dynamics in time, we apply integration via an ODE numerical solver [18].

Overall flow. Combined altogether, our components approximate the data flow Φ in Eq. (1):

$$\forall (t, \tau), \qquad (v_t, \tau) \mapsto D_\phi\Big(E_\varphi(v_t) + \int_t^{t+\tau} f_\psi(\alpha_{\tau'})\,\mathrm{d}\tau'\Big) \quad \text{where } \alpha_t = E_\varphi(v_t). \tag{5}$$

To summarize, DINo defines a time-continuous latent temporal model with a space-continuous emission function D_ϕ, combining the flexibility of space and time continuity. This is fully novel to our knowledge, as prior latent approaches are discretized (cf. [14] for state-space models).

3.3 Training

Given these three components (ENC), (DEC), (DYN), we now present their training procedure, illustrated in Figure 2 (right). We use a fast and simple two-stage optimization, close to recent works in video prediction [48], and provide implementation details in Section F. Given the train sequences $\mathcal{D}$, we first apply auto-decoding across times to obtain the corresponding latent vectors $\alpha_\mathcal{T} = \{\alpha_t^v\}_{t\in\mathcal{T},v\in\mathcal{D}}$, as well as the decoder parameters ϕ. We then learn the parameters of the dynamics model ψ by modeling the latent flow over α_t^v for each $v \in \mathcal{D}$. We detail this procedure in Section F, which can be formalized as a bi-level optimization problem solved in parallel:

$$\begin{aligned} \min_\psi \quad & \ell_{\text{dyn}}(\psi, \alpha_\mathcal{T}) \triangleq \mathbb{E}_{v\in\mathcal{D},t\in\mathcal{T}} \left\| \alpha_t^v - \left(\alpha_0^v + \int_0^t f_\psi(\alpha_\tau^v)\,\mathrm{d}\tau \right) \right\|_2^2 \\ \text{s.t. } \alpha_\mathcal{T}, \phi = \arg\min_{\alpha_\mathcal{T},\phi} \quad & \ell_{\text{dec}}(\phi, \alpha_\mathcal{T}) \triangleq \mathbb{E}_{v\in\mathcal{D},x\in\mathcal{X}_{\text{tr}},t\in\mathcal{T}} \left\| v_t(x) - D_\phi(\alpha_t^v)(x) \right\|_2^2. \end{aligned} \tag{6}$$

3.4 Decoder Implementation via Amplitude-Modulated INRs

We now specify our implementation of decoder D_ϕ in Eq. (4). This includes the definition of the INR architecture I_θ and of the hypernetwork h_ϕ. We introduce for the latter a new method called amplitude modulation, which implements a space-time separation of variables.

I_θ as FourierNet. We implement I_θ as a FourierNet, a state-of-the-art INR architecture, which instantiates a Multiplicative Filter Network (MFN, [13]). A FourierNet relies on the recursion in Eq. (7), where $x \in \Omega$ is an input spatial location, $z^{(l)}(x)$ is the hidden feature vector at layer l for x and $s_{\omega^{(l)}}(x) = [\cos(\omega^{(l)}x), \sin(\omega^{(l)}x)]$ is a Fourier basis, with a frequency matrix $\omega^{(l)}$:

$$\begin{cases} z^{(0)}(x) = s_{\omega^{(0)}}(x), \\ z^{(l)}(x) = \left(W^{(l-1)} z^{(l-1)}(x) + b^{(l-1)} \right) \odot s_{\omega^{(l)}}(x) \quad \text{for } l \in [\![1, L-1]\!], \\ z^{(L)}(x) = W^{(L-1)} z^{(L-1)}(x) + b^{(L-1)} \end{cases} \tag{7}$$

where we fix $W^{(0)} = 0$, $b^{(0)} = 1$, $s_{\omega^{(0)}}(x) = x$. Denoting $W = [W^{(l)}]_{l=1}^{L-1}, b = [b^{(l)}]_{l=1}^{L-1}, \omega = [\omega^{(l)}]_{l=1}^{L-1}$, we fit a FourierNet to an input function v observed on a grid $\mathcal{X}$ by learning $\{W, b, \omega\}$ s.t. $\forall x \in \mathcal{X}, z^{(L)}(x) = v(x)$. In practice, we observe that fixing ω uniformly sampled in a given interval (cf. Section F) performs similarly to learning them, so we exclude them from training parameters.

We leverage the interpretability of FourierNets to separate time and space via amplitude modulation. [13] shows that $\exists M \gg L \in \mathbb{N}, \exists\{c_j^{(m)}\}_{m=1}^M$ coefficients that depend individually on $\{W, b\}$ and $\exists\{\gamma^{(m)}\}_{m=1}^M$ parameters that depend individually on those of the filters ω s.t. the j^{th} dimension of $z^{(L)}(x)$ can be expressed as:

$$z_j^{(L)}(x) = \sum_{m=1}^M c_j^{(m)} s_{\gamma^{(m)}}(x) + \text{bias}. \tag{8}$$

Eq. (8) involves a basis of spatial functions $\{s_{\gamma^{(m)}}\}_{m=1}^M$ evaluated at x and their amplitudes $\{c_j^{(m)}\}_{m=1}^M$. Note that Eq. (8) can be extended to other choices of $s_{\omega^{(l)}}$ [13].

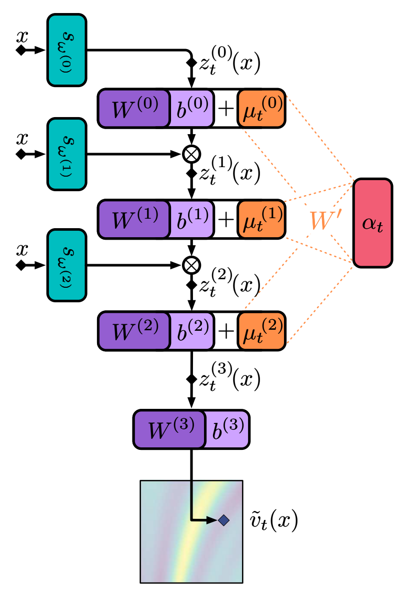


Fig. 4: Amplitude modulation – Eq. (9)

h as amplitude modulation. h generates the INR's parameters θ_t given α_t to model a target function v_t. We implement h as elementwise shift and scale transformations [FiLM, 36] of the linear layers parameters W, b (excluding those of the filters ω). Then in Eq. (8) amplitudes $c_j^{(m)}$ only depend on time while the basis functions $s_{\gamma^{(m)}}$ only depend on space: this corresponds to separation of variable [25]. We call our technique amplitude modulation. In practice, as [12], we consider latent shift mappings: Figure 4 and Eq. (9) extend Eq. (7) with a shift term $\mu_t^{(l-1)}$ at each layer l, $\mu_t^{(l-1)} \triangleq W'^{(l-1)}\alpha_t$, where $W' = [W'^{(l-1)}]_{l=1}^{L-1}$ is a weight matrix:

$$z_t^{(l)}(x) = \left(W^{(l-1)} z_t^{(l-1)}(x) + b^{(l-1)} + \mu_t^{(l-1)} \right) \odot s_{\omega^{(l)}}(x). \tag{9}$$

Table 2: **Space and time extrapolation.** Train and test observation grids are equal and subsampled from an uniform 64×64 grid, used for inference. We report MSE (↓) on the inference time interval $\mathcal{T}'$, divided within training horizon (*In-t*, $\mathcal{T}$) and beyond (*Out-t*, outside $\mathcal{T}$) across subsampling ratios.

Model		Navier-Stokes				Wave			
		Train		Test		Train		Test	
		In-t	Out-t	In-t	Out-t	In-t	Out-t	In-t	Out-t
$s = 5\%$ subsampling ratio									
Discrete	I-MP-PDE	8.154E−3	8.166E−3	7.926E−3	8.225E−3	7.055E−4	7.097E−4	1.138E−3	1.116E−3
Operator	DeepONet	3.330E−3	7.370E−3	1.346E−2	1.408E−2	8.331E−4	9.295E−3	1.692E−2	3.256E−2
INR	SIREN	8.741E−3	1.767E−1	4.303E−2	2.126E−1	2.738E−3	1.818E−2	3.339E−2	6.964E−2
INR	DINo	**1.029E−3**	**1.655E−3**	**1.326E−3**	**1.813E−3**	**4.088E−5**	**4.121E−5**	**6.415E−5**	**7.392E−5**
$s = 100\%$ subsampling ratio									
Discrete	CNODE	2.319E−2	9.652E−2	2.305E−2	1.143E−1	2.337E−5	5.280E−4	3.057E−5	7.288E−4
Discrete	MP-PDE	1.140E−4	5.500E−4	**1.785E−4**	5.856E−4	**1.718E−7**	1.993E−5	**9.256E−7**	4.261E−5
Operator	MNO	**3.190E−5**	8.678E−4	2.763E−4	8.946E−4	9.381E−6	4.890E−3	1.993E−4	6.128E−3
Operator	DeepONet	1.375E−3	6.573E−3	9.704E−3	1.244E−2	6.431E−4	1.293E−2	1.847E−2	3.317E−2
INR	SIREN	1.066E−3	4.336E−1	3.874E−1	1.037E0	3.674E−4	9.956E−3	3.013E−2	7.842E−2
INR	MFN	1.651E−3	1.037E0	2.106E−1	1.059E0	1.408E−4	1.763E−1	4.735E−3	2.274E−1
INR	DINo (no sep.)	3.235E−4	1.593E−3	7.850E−4	1.889E−3	2.641E−6	4.081E−5	5.977E−5	2.979E−4
INR	DINo	8.339E−5	**3.115E−4**	2.092E−4	**4.311E−4**	3.309E−6	**3.506E−6**	9.495E−6	**9.946E−6**

The INR's parameters are defined as $h_\phi(\alpha_t) = \{W; b + W'\alpha_t; \omega\}$ where $\phi = \{W, b, W', \omega\}$ are h's parameters. Thus, amplitude modulation separates time and space. We show in Table 5 that it significantly improves performance, particularly time extrapolation.

4 Experiments

We assess the spatiotemporal versatility of DINo, following Section 2. We introduce our experimental setting (Section 4.1), which includes several challenging PDEs, state-of-the-art baselines and forecasting tasks. Then, we analyze the experimental results (Section 4.2). To reproduce our results, our code is publicly available at `https://github.com/mkirchmeyer/DINo`.

4.1 Experimental Setting

Datasets. We consider the following PDEs defined over a spatial domain Ω, with further details in Section E. • **2D Wave equation** (*Wave*) is a second-order PDE $\frac{\partial^2 u}{\partial t^2} = c^2 \Delta u$. u is the displacement w.r.t. the rest position and c is the wave traveling speed. We consider its first-order form, so that $v_t = (u_t, \frac{\partial u_t}{\partial t})$ has a two-dimensional output ($n = 2$). • **2D Navier Stokes** ([43], *Navier-Stokes*) corresponds to an incompressible fluid dynamics $\frac{\mathrm{d}v}{\mathrm{d}t} = -u\nabla v + \nu \Delta v + f, v = \nabla \times u, \nabla u = 0$, where u is the velocity field and v the vorticity. ν is the viscosity and f is a constant forcing term; $n = 1$. • **3D Spherical shallow water** ([16], *Shallow-Water*): it involves the vorticity w, tangent to the sphere's surface, and the thickness of the fluid h. The input is $v_t = (w_t, h_t)$ ($n = 2$).

Baselines. We reimplement representative models from Section A and Table 1 and adapt them to our multi-dimensional datasets. • **CNODE** [1] combines a CNN and ODE solver to handle regular grids. • **MP-PDE** [4] uses a GNN to handle free-formed grids, yet cannot predict outside the observed grid. We developed an interpolative extension, **I-MP-PDE**, to handle this limitation; training is done on the bicubic interpolation of the observed grid. • **MNO** [27]: an autoregressive version of FNO [28] for regular grids; MNO can be evaluated on new uniform grids. • **DeepONet** [31], considered autoregressively [45] where we remove time from the trunk net's input. DeepONet can be evaluated on new spatial locations without interpolation. • **SIREN** [41] and **MFN** [13]: two INR methods extended to our setting. We consider an agnostic setting, i.e. without the knowledge of the PDE and perform sequence conditioning to generalize to more than a trajectory. This is achieved by learning a latent vector with auto-decoding; it is then concatenated to the spatial coordinates.

Tasks. We evaluate models on various forecasting tasks which combine the evaluation scenarios of Section 2. Performance is measured by the prediction Mean Squared Error (MSE) given only an initial condition. • **Space and time extrapolation.** We consider a uniform grid $\mathcal{X}'$ for inference. Training is performed on different observations grids $\mathcal{X}_{\text{tr}}$ subsampled from $\mathcal{X}'$ with different ratios,

Table 3: **Flexibility w.r.t. input grid.** Observed test / train grid differ ($\mathcal{X}_{ts} \neq \mathcal{X}_{tr}$). We report *test* MSE ($\downarrow$) for *Navier-Stokes* on $\mathcal{X}' = \mathcal{X}_{ts}$ (*In-s*). **Green** **Yellow** **Red** mean excellent, good, poor MSE.

(a) **Generalization across grids**: $\mathcal{X}_{tr}, \mathcal{X}_{ts}$ are subsampled with different ratios $s_{tr} \neq s_{ts}$ among $\{5, 50, 100\}\%$ from the same uniform 64×64 grid.

Subsampling / Train ↓	Test→	$s_{ts} = 5\%$ In-t	$s_{ts} = 5\%$ Out-t	$s_{ts} = 50\%$ In-t	$s_{ts} = 50\%$ Out-t	$s_{ts} = 100\%$ In-t	$s_{ts} = 100\%$ Out-t
$s_{tr} = 5\%$	MP-PDE	1.330E−1	3.852E−1	1.859E−1	6.680E−1	2.105E−1	7.120E−1
	DINO	**1.494E−3**	**2.291E−3**	**1.257E−3**	**1.883E−3**	**1.287E−3**	**1.947E−3**
$s_{tr} = 50\%$	MP-PDE	4.494E−2	9.403E−2	4.793E−3	1.997E−2	6.330E−3	3.712E−2
	DINO	**2.470E−4**	**4.697E−4**	**2.073E−4**	**4.284E−4**	**2.058E−4**	**4.361E−4**
$s_{tr} = 100\%$	MP-PDE	1.358E−1	3.355E−1	1.182E−2	2.664E−2	**1.785E−4**	5.856E−4
	DINO	**2.495E−4**	**4.805E−4**	**2.109E−4**	**4.325E−4**	2.092E−4	**4.311E−4**

(b) **Generalization across resolutions**: $\mathcal{X}_{ts}$ (resp. $\mathcal{X}_{tr}$) are subsampled at the same ratio $s \in \{5, 100\}\%$ from different uniform grids with resolution $r_{ts} \in \{32, 64, 256\}$ (resp. $r_{tr} = 64$).

Subsampling ↓	Test resolution →	$r_{ts} = 32$ - $\mathcal{X}_{ts} \neq \mathcal{X}_{tr}$ In-t	$r_{ts} = 32$ - $\mathcal{X}_{ts} \neq \mathcal{X}_{tr}$ Out-t	$r_{ts} = 64$ - $\mathcal{X}_{ts} = \mathcal{X}_{tr}$ In-t	$r_{ts} = 64$ - $\mathcal{X}_{ts} = \mathcal{X}_{tr}$ Out-t	$r_{ts} = 256$ - $\mathcal{X}_{ts} \neq \mathcal{X}_{tr}$ In-t	$r_{ts} = 256$ - $\mathcal{X}_{ts} \neq \mathcal{X}_{tr}$ Out-t
$s = 5\%$	MP-PDE	3.209E−1	6.472E−1	**2.465E−4**	1.105E−3	2.239E−1	8.253E−1
	DINO	**5.308E−3**	**9.544E−3**	2.533E−4	**8.832E−4**	**1.991E−3**	**2.942E−3**
$s = 100\%$	MNO	4.547E−3	9.281E−3	**1.277E−4**	8.525E−4	2.174E−3	4.975E−3
	MP-PDE	4.194E−2	9.109E−2	1.597E−4	6.483E−4	4.648E−2	1.381E−1
	DINO	**2.321E−4**	**6.386E−4**	2.320E−4	**6.385E−4**	**2.322E−4**	**6.385E−4**

$s \in \{5\%, 25\%, 50\%, 100\%\}$ where $s = 100\%$ corresponds to the full inference grid, i.e. $\mathcal{X}_{tr} = \mathcal{X}'$. In this setting, we consider that all trajectories (*train* and *test*) share the same observation grid $\mathcal{X}_{tr} = \mathcal{X}_{ts}$. We evaluate MSE error on $\mathcal{X}'$ over the train time interval (*In-t*) and beyond (*Out-t*) at each subsampling ratio. • **Flexibility w.r.t. input grid.** We vary the test observation grid, i.e. $\mathcal{X}_{ts} \neq \mathcal{X}_{tr}$ and perform inference on $\mathcal{X}' = \mathcal{X}_{ts}$, i.e. on the test observation grid (*In-s*) under two settings: ▷ **Generalizing across grids:** $\mathcal{X}_{tr}, \mathcal{X}_{ts}$ are subsampled differently from the same uniform grid; s_{tr} (resp. s_{ts}) is the train (resp. test) subsampling ratio. ▷ **Generalizing across resolutions:** $\mathcal{X}_{tr}, \mathcal{X}_{ts}$ are subsampled with the same ratio s from two uniform grids with different resolutions; the train resolution is fixed to $r_{tr} = 64$ while we vary the test resolution $r_{ts} \in \{32, 64, 256\}$. • **Data on manifold.** combines several evaluation scenarios for a PDE on a sphere. • **Finer time resolution.** considers a finer inference time grid than the train one; we present this experiment in Section B.

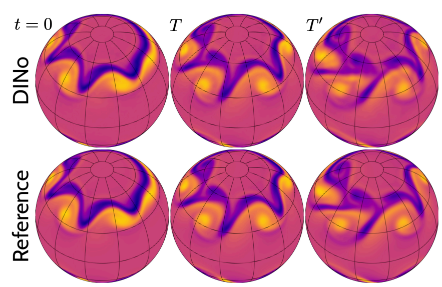
4.2 Results

Space and time extrapolation. We report prediction MSE in Table 2 for varying subsampling ratios $s \in \{5, 100\}\%$ on *Navier-Stokes* and *Wave*. Section C provides a fine-grained evaluation inside the train observation grid (*In-s*) or outside (*Out-s*) and reports additionally the results for $s \in \{25, 50\}\%$. We visualize some predictions in Section D. DINO is compared to all baselines when $s = 100\%$, i.e. $\mathcal{X}' = \mathcal{X}_{tr} = \mathcal{X}_{ts}$, and otherwise it is compared only to models which handle irregular grids and prediction at arbitrary spatial locations (DeepONet, SIREN, MFN, I-MP-PDE). • **General analysis.** We observe that all models degrade when the subsampling ratio s decreases. DINO performs competitively overall: it achieves the best *Out-t* performance on all subsampling settings, it outperforms all the baselines on low subsampling ratios and performs comparably to the competitive discretized alternatives (MP-PDE, CNODE) and operator (MNO) when $s = 100\%$, i.e. when observation and inference grids are equal. Note that this fully observed setting is favorable for CNODE, MP-PDE and MNO, designed to perform inference on the observation grid. This can be seen in Table 2, where DINO is slightly outperformed only for few settings. MP-PDE is significantly better only on *Wave* for *In-t*. Overall, CNNs and GNNs exhibit good performance for spatially local dynamics like *Wave*, while INRs (like DINO) and MNO are more adapted to global dynamics like *Navier-Stokes*. • **Analysis per model.** MP-PDE is the most competitive baseline across datasets as it combines a strong and flexible encoder (GNNs) to a good dynamics model; however, it cannot predict outside the observation grid (*Out-s*). To keep a strong competitor, we extend this baseline into its interpolative version I-MP-PDE on subsampled settings. I-MP-PDE is competitive for high subsampling ratios, e.g. $s \in \{50\%, 100\%\}$, but underperforms w.r.t. DINO at lower subsampling

ratios due to the accumulated interpolation error. MNO is a competitive baseline on *Navier-Stokes*, performing on par with MP-PDE and DINo inside the training horizon (*In-t*); its performance on *Out-t* degrades more significantly compared to other models, especially DINo. DeepONet is more flexible than MP-PDE as it can predict at arbitrary locations. As no interpolation error is introduced, it outperforms I-MP-PDE for $s = 5\%$ on *train* data. Yet, we observe that it underperforms especially on *Out-t* w.r.t. its alternatives. Finally, we observe that SIREN and MFN fit correctly the train horizon *In-t* on *train*, yet generalize poorly outside this horizon *Out-t* or on new initial conditions (*test*). This is in accordance with our analysis of Section A; we highlight that this is not the case for DINo which extrapolates temporally and generalizes to new initial conditions thanks to its sequential modeling of the flow. Thus, DINo *is currently the state-of-the-art INR model for spatiotemporal data*.
• **Modulation.** We observe that modulating both amplitudes and frequencies (row "DINo (no sep.)" in Table 2) degrades performance w.r.t. DINo (row DINo in Table 2) that only modulates amplitudes. Amplitude modulation enables long temporal extrapolation and reduces the number of parameters. Hence, as opposed to DINo (no sep.) which is outperformed by some baselines, time-space variable separation in DINo is an essential ingredient of the model to reach state-of-the-art levels.

Flexibility w.r.t. input grid. We consider in Table 3 *Navier-Stokes* and compare DINo to the most competitive baselines, MP-PDE and MNO (with $s = 100\%$ subsampling ratio). • **Generalizing across grids.** In Table 3a, we consider that the test observation grid $\mathcal{X}_{\text{ts}}$ is different from the train one $\mathcal{X}_{\text{tr}}$. This occurs when sensors differ between two observed trajectories. We vary the subsampling ratio for the train observation grid s_{tr} and the test one s_{ts}. We report *test* MSE on new grids $\mathcal{X}' = \mathcal{X}_{\text{ts}}$. DINo is very robust to changing grids between *train* and *test*, while MP-PDE's performance degrades, especially for low subsampling ratios, e.g. 5%. For reference, we report in Section C, Table 6 (col. 3), the performance when $\mathcal{X}' = \mathcal{X}_{\text{tr}}$, where MP-PDE is substantially better. • **Generalizing across spatial resolutions.** In Table 3b we vary the test resolution r_{ts}. We train at a resolution $r_{\text{tr}} = 64$ and perform inference at resolutions $r_{\text{ts}} \in \{32, 64, 256\}$. For that, we build a high-fidelity 256×256 simulation dataset and downscale it to obtain the other resolutions. We observe that DINo's performance is the stablest across resolutions in the uniform or irregular setting. MNO is also relatively stable but is only applicable to uniform grids while MP-PDE is particularly brittle, especially for a 5% ratio.

Data on manifold. We consider in Figure 5 *Shallow-Water* in a super-resolution setting: test resolution is twice the train one, close to weather prediction applications. We observe an irregular 3D Euclidean coordinate grid $\mathcal{X}_{\text{tr}} = \mathcal{X}_{\text{ts}} \subset \mathbb{R}^3$ shared for *train* and *test*. It samples uniformly Euclidean positions on the sphere, via the quasi-uniform skipped latitude-longitude grid [46]. We predict the PDE on *test* trajectories with a conventional latitude-longitude inference grid $\mathcal{X}'$. At Earth scale, $\mathcal{X}_{\text{tr}}$ corresponds to a resolution of about 300 km, and $\mathcal{X}'$ to 150 km. DINo significantly outperforms I-MP-PDE, making it a viable candidate for this complex setting.



Model	In-t	Out-t
I-MP-PDE	1.908E−3	7.240E−3
DINo	**1.063E−4**	**6.466E−4**

Figure 5: **Data on manifold.** DINo's *Shallow-Water* superresolution *test* prediction (top) against the reference (middle); *test* MSE comparison (↓) (bottom).

5 Conclusion

We propose DINo, a novel space-time continuous data-driven PDE forecaster. DINo handles settings encountered in many applications, where existing methods fail. We assess its extensive spatiotemporal extrapolation on several PDEs and its generalization to unseen sparse irregular meshes and resolutions. Its competitive results over recent neural PDE forecasters make it a strong alternative for real-world free-formed spatiotemporal conditions. There are many promising extensions of DINo, for example improving its generalization to new system parameters [22] or to new boundary conditions.

Acknowledgement

We thank Emmanuel de Bézenac and Jérémie Donà for helpful insights and discussions on this project. We also acknowledge financial support from DL4CLIM (ANR-19-CHIA-0018-01) and DEEPNUM (ANR-21-CE23-0017-02) ANR projects.

References

[1] Ibrahim Ayed, Emmanuel de Bézenac, Arthur Pajot, and Patrick Gallinari. Learning the spatio-temporal dynamics of physical processes from partial observations. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3232–3236, May 2020. (cit. on pp. 1 and 6)

[2] Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. In Corinna Cortes, Neil D. Lawrence, Daniel D. Lee, Masashi Sugiyama, and Roman Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28, pages 1171—1179. Curran Associates, Inc., 2015. (cit. on p. 19)

[3] Marsha J. Berger and Joseph Oliger. Adaptive mesh refinement for hyperbolic partial differential equations. *Journal of Computational Physics*, 53(3):484–512, March 1984. (cit. on p. 2)

[4] Johannes Brandstetter, Daniel E. Worrall, and Max Welling. Message passing neural PDE solvers. In *International Conference on Learning Representations*, 2022. (cit. on pp. 1, 2, 6, 13, and 20)

[5] Steven L. Brunton and J. Nathan Kutz. *Data-driven science and engineering: Machine learning, dynamical systems, and control*. Cambridge University Press, 2022. (cit. on p. 1)

[6] Keaton J. Burns, Geoffrey M. Vasil, Jeffrey S. Oishi, Daniel Lecoanet, and Benjamin P. Brown. Dedalus: A flexible framework for numerical simulations with spectral methods. *Physical Review Research*, 2(2), April 2020. (cit. on p. 19)

[7] Sangwon Chae, Joonhyeok Shin, Sungjun Kwon, Sangmok Lee, Sungwon Kang, and Donghyun Lee. PM10 and PM2.5 real-time prediction models using an interpolated convolutional neural network. *Scientific Reports*, 11, 2021. (cit. on p. 13)

[8] Hao Chen, Bo He, Hanyu Wang, Yixuan Ren, Ser Nam Lim, and Abhinav Shrivastava. NeRV: Neural representations for videos. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann Dauphin, Percy Liang, and Jenn Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 21557–21568. Curran Associates, Inc., 2021. (cit. on p. 13)

[9] Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, and David Duvenaud. Neural ordinary differential equations. In Samy Bengio, Hanna Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31, pages 6572–6583. Curran Associates, Inc., 2018. (cit. on pp. 2 and 20)

[10] Filipe De Avila Belbute-Peres, Thomas Economon, and Zico Kolter. Combining differentiable PDE solvers and graph neural networks for fluid flow prediction. In Hal Daumé, III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 2402–2411. PMLR, July 2020. (cit. on p. 1)

[11] Emmanuel de Bézenac, Arthur Pajot, and Patrick Gallinari. Deep learning for physical processes: Incorporating prior scientific knowledge. In *International Conference on Learning Representations*, 2018. (cit. on p. 13)

[12] Emilien Dupont, Hyunjik Kim, S. M. Ali Eslami, Danilo Jimenez Rezende, and Dan Rosenbaum. From data to functa: Your data point is a function and you can treat it like one. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvári, Gang Niu, and Sivan Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 5694–5725. PMLR, July 2022. (cit. on p. 5)

[13] Rizal Fathony, Anit Kumar Sahu, Devin Willmott, and J. Zico Kolter. Multiplicative filter networks. In *International Conference on Learning Representations*, 2021. (cit. on pp. 5, 6, and 13)

[14] Marco Fraccaro. *Deep Latent Variable Models for Sequential Data*. PhD thesis, Danmarks Tekniske Universitet, 2018. (cit. on p. 4)

[15] Stefania Fresca, Andrea Manzoni, Luca Dedè, and Alfio Quarteroni. Deep learning-based reduced order models in cardiac electrophysiology. *PLoS ONE*, 15(10), October 2020. (cit. on p. 13)

[16] Joseph Galewsky, Richard K. Scott, and Lorenzo M. Polvani. An initial-value problem for testing numerical models of the global shallow-water equations. *Tellus A: Dynamic Meteorology and Oceanography*, 56(5):429–440, 2004. (cit. on pp. 6, 18, and 19)

[17] David Ha, Andrew M. Dai, and Quoc V. Le. HyperNetworks. In *International Conference on Learning Representations*, 2017. (cit. on p. 4)

[18] Ernst Hairer, Syvert P. Nørsett, and Gerhard Wanner. *Solving Ordinary Differential Equations I*. Springer Series in Computational Mathematics. Springer Berlin Heidelberg, Berlin - Heidelberg, Germany, 2nd edition edition, 1993. (cit. on p. 4)

[19] Valerii Iakovlev, Markus Heinonen, and Harri Lähdesmäki. Learning continuous-time PDEs from sparse data with graph neural networks. In *International Conference on Learning Representations*, 2021. (cit. on p. 13)

[20] Hyunjik Kim, Andriy Mnih, Jonathan Schwarz, Marta Garnelo, Ali Eslami, Dan Rosenbaum, Oriol Vinyals, and Yee Whye Teh. Attentive neural processes. In *International Conference on Learning Representations*, 2019. (cit. on p. 4)

[21] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015. (cit. on p. 20)

[22] Matthieu Kirchmeyer, Yuan Yin, Jérémie Donà, Nicolas Baskiotis, Alain Rakotomamonjy, and Patrick Gallinari. Generalizing to new physical systems via context-informed dynamics model. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvári, Gang Niu, and Sivan Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 11283–11301. PMLR, July 2022. (cit. on p. 8)

[23] Dmitrii Kochkov, Jamie A. Smith, Ayya Alieva, Qing Wang, Michael P. Brenner, and Stephan Hoyer. Machine learning-accelerated computational fluid dynamics. *Proceedings of the National Academy of Sciences*, 118(21), 2021. (cit. on p. 1)

[24] Nikola Kovachki, Zongyi Li, Burigede Liu, Kamyar Azizzadenesheli, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Neural operator: Learning maps between function spaces. *arXiv preprint arXiv:2108.08481*, 2021. (cit. on p. 13)

[25] Hervé Le Dret and Brigitte Lucquin. *Partial Differential Equations: Modeling, Analysis and Numerical Approximation*, chapter The Heat Equation, pages 219–251. International Series of Numerical Mathematics. Springer International Publishing, Cham, Switzerland, 2016. (cit. on p. 5)

[26] Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Multipole graph neural operator for parametric partial differential equations. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 6755–6766. Curran Associates, Inc., 2020. (cit. on pp. 1 and 13)

[27] Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Markov neural operators for learning chaotic systems. *arXiv preprint arXiv:2106.06898*, 2021. (cit. on pp. 2, 6, and 13)

[28] Zongyi Li, Nikola Borislavov Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. In *International Conference on Learning Representations*, 2021. (cit. on pp. 1, 6, 13, 18, and 20)

[29] Zongyi Li, Daniel Zhengyu Huang, Burigede Liu, and Anima Anandkumar. Fourier neural operator with learned deformations for PDEs on general geometries. *arXiv preprint arXiv:2207.05209*, 2022. (cit. on p. 13)

[30] Zichao Long, Yiping Lu, Xianzhong Ma, and Bin Dong. PDE-Net: Learning PDEs from data. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 3208–3216, Stockholmsmässan, Stockholm, Sweden, July 2018. PMLR. (cit. on p. 13)

[31] Lu Lu, Pengzhan Jin, Guofei Pang, Zhongqiang Zhang, and George Em Karniadakis. Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators. *Nature Machine Intelligence*, 3:218–229, 2021. (cit. on pp. 1, 2, 6, and 13)

[32] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. NeRF: Representing scenes as neural radiance fields for view synthesis. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *The European Conference on Computer Vision (ECCV)*, pages 405–421, Cham, Switzerland, 2020. Springer International Publishing. (cit. on p. 1)

[33] Peter J. Olver. *Introduction to partial differential equations*. Undergraduate Texts in Mathematics. Springer Cham, 2014. (cit. on p. 1)

[34] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. DeepSDF: Learning continuous signed distance functions for shape representation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 165–174, June 2019. (cit. on p. 4)

[35] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An imperative style, high-performance deep learning library. In Hanna Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché Buc, Emily Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32, pages 8026–8037. Curran Associates, Inc., 2019. (cit. on p. 19)

[36] Ethan Perez, Florian Strub, Harm de Vries, Vincent Dumoulin, and Aaron Courville. FiLM: Visual reasoning with a general conditioning layer. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32(1):3942–3951, April 2018. (cit. on p. 5)

[37] Tobias Pfaff, Meire Fortunato, Alvaro Sanchez-Gonzalez, and Peter W. Battaglia. Learning mesh-based simulation with graph networks. In *International Conference on Learning Representations*, 2021. (cit. on pp. 1 and 13)

[38] Michael Prasthofer, Tim De Ryck, and Siddhartha Mishra. Variable-input deep operator networks. *arXiv preprint arXiv:2205.11404*, 2022. (cit. on p. 1)

[39] Maziar Raissi, Paris Perdikaris, and George Em Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019. (cit. on pp. 1, 2, and 13)

[40] Justin Sirignano and Konstantinos Spiliopoulos. DGM: A deep learning algorithm for solving partial differential equations. *Journal of Computational Physics*, 375:1339–1364, 2018. (cit. on p. 1)

[41] Vincent Sitzmann, Julien N. P. Martel, Alexander W. Bergman, David B. Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 7462–7473. Curran Associates, Inc., 2020. (cit. on pp. 1, 6, 13, and 20)

[42] Ivan Skorokhodov, Sergey Tulyakov, and Mohamed Elhoseiny. StyleGAN-V: A continuous video generator with the price, image quality and perks of StyleGAN2. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3626–3636, June 2022. (cit. on p. 13)

[43] George Gabriel Stokes. On the effect of the internal friction of fluids on the motion of pendulums. *Transactions of the Cambridge Philosophical Society*, 9(2):8–106, 1851. (cit. on pp. 6 and 17)

[44] Matthew Tancik, Pratul P. Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan T. Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 7537–7547. Curran Associates, Inc., 2020. (cit. on pp. 1 and 13)

[45] Sifan Wang and Paris Perdikaris. Long-time integration of parametric evolution equations with physics-informed DeepONets. *arXiv preprint arXiv:2106.05384*, 2021. (cit. on p. 6)

[46] Hilary Weller, John Thuburn, and Colin J. Cotter. Computational modes and grid imprinting on five quasi-uniform spherical C grids. *Monthly Weather Review*, 140(8):2734–2755, 2012. (cit. on p. 8)

[47] Jared Willard, Xiaowei Jia, Shaoming Xu, Michael Steinbach, and Vipin Kumar. Integrating scientific knowledge with machine learning for engineering and environmental systems. *ACM Computing Surveys*, January 2022. (cit. on p. 1)

[48] Wilson Yan, Yunzhi Zhang, Pieter Abbeel, and Aravind Srinivas. VideoGPT: Video generation using VQ-VAE and transformers. *arXiv preprint arXiv:2104.10157*, 2021. (cit. on p. 5)

[49] Yuan Yin, Vincent Le Guen, Jérémie Donà, Emmanuel de Bézenac, Ibrahim Ayed, Nicolas Thome, and Patrick Gallinari. Augmenting physical models with deep networks for complex dynamics forecasting. *Journal of Statistical Mechanics: Theory and Experiment*, December 2021. (cit. on pp. 1 and 13)

[50] Sihyun Yu, Jihoon Tack, Sangwoo Mo, Hyunsu Kim, Junho Kim, Jung-Woo Ha, and Jinwoo Shin. Generating videos with dynamics-aware implicit generative adversarial networks. In *International Conference on Learning Representations*, 2022. (cit. on p. 13)

[51] Laure Zanna and Thomas Bolton. *Deep Learning of Unresolved Turbulent Ocean Processes in Climate Models*, chapter 20, pages 298–306. John Wiley & Sons, Ltd, 2021. (cit. on p. 2)

Appendices

A Related Work

We review current data-driven approaches for PDE modeling and the representative methods listed in Table 1. We express the forecasting rule using the notations in Eq. (1): t is an arbitrary time; τ is an arbitrary time interval; δt is a fixed, predetermined time interval (as a model hyperparameter).

Sequential discretized models. Most sequential dynamics models are learned on a fixed observed grid $\mathcal{X}_{\text{tr}}$ and use discretized models, e.g. CNNs or GNNs, to process the observations. CNNs require observations on a regular grid but can be extended to irregular grids through interpolation [7]. GNNs are more flexible as they handle irregular grids, at an additional memory and computational cost. Yet, prediction on new grids $\mathcal{X}' \neq \mathcal{X}_{\text{tr}}$ fails experimentally for both CNNs and GNNs, as these discretized models are biased towards the training grid $\mathcal{X}_{\text{tr}}$, as we show in Section 4. We distinguish two types of temporal models which both extrapolate beyond the train horizon due to their sequential nature.

- Autoregressive models $\boldsymbol{v_t|_{\mathcal{X}} \mapsto v_{t+\delta t}|_{\mathcal{X}}}$ [30, 11, 37, 4]. These models predict the sequence from t only at fixed time increments δt and not in between.
- Time-continuous extensions using numerical solvers $\boldsymbol{(v_t|_{\mathcal{X}}, \tau) \mapsto v_{t+\tau}|_{\mathcal{X}}}$ [49, 19] solve this limitation as they provide a prediction at arbitrary times, thus remove dependency on the time discretization.

Operator learning. Recently, operator-based models aim at finding a parameterized mapping between functions. They define in theory space-continuous models. First, neural operators [24] attempt to replace standard convolution with continuous alternatives. Fourier Neural Operator (FNO, [28]) applies convolution in the spectral domain via Fast Fourier Transformation (FFT). Graph Neural Operator (GNO, [26]) performs convolution on a local interaction grid described by a graph. Second, DeepONet [31] uses a coordinate-based neural network to output a prediction at arbitrary time and space locations given a function observed on a fixed grid. Three types of temporal models were used for operators with some limitations.

- The standard approach, $\boldsymbol{v_0 \mapsto v_t}$, models the output at a given time $t \in [0, T]$ within the train horizon [26].
- A sequential extension, $\boldsymbol{v_t \mapsto v_{t+\delta t}}$, was proposed in [27].
- Finally, a time-continuous version $\boldsymbol{v_0 \mapsto (t \in [0, T] \mapsto v_t)}$ in DeepONet propose a solution at arbitrary time and space locations.

The first and third approaches are not designed to generalize beyond the train horizon, i.e. when $t > T$ as they are not sequential. The second solves this limitation but is only able to predict solutions from t at fixed time increments of δt and not in-between. Furthermore, all existing approaches make restrictive assumptions on the space discretization. They lack flexibility when encoding spatial observations: FNO is limited to uniform Cartesian observation grids due to FFT; GNO does not adapt well to changing observation grids as for the GNN-based models in the previous paragraph; DeepONet is limited to input observations on fixed observation locations – the latter are chosen at random spatial positions but should remain fixed throughout training and testing. Concurrent to our work, [29] extends Markov FNO [28, 27] and circumvents these issues by embedding spatial observations in a Euclidean space; however, its spatial flexibility remains limited: it does not handle arbitrary topologies, it has a quadratic complexity in the number of points, and, like its predecessors, cannot be evaluated at unobserved locations in the conditioning frame.

Spatiotemporal INRs. Another class of models is based on coordinate-based NNs, called Implicit Neural Representations [INRs, 41, 13, 44]. These space-continuous models share a similar objective as operators, despite constituting a separate research field. INRs for spatiotemporal data take time as an input along spatial coordinates. Physics-informed neural networks (PINNs, [39]) use this formulation to solve PDEs, yet are limited to a single known differential equation and a set of initial and boundary conditions. [15] proposes an agnostic INR approach to build reduced order models for electrophysiology. Extensions for multi-sequence learning, e.g. for video generation [50, 42] or compression [8], learn a latent conditioning variable from an initial condition v_0, i.e. take the form

Table 4: **Finer time resolution.** *Test* MSE (↓) under $\mathcal{T}'$ for *Navier-Stokes*.

Model	In-t	Out-t
I-DINo (linear)	3.459E−4	5.598E−4
I-DINo (quadratic)	2.165E−4	4.473E−4
DINo (ODE solve)	**2.151E−4**	**4.388E−4**

$\boldsymbol{v_0 \mapsto (t \in [0, T] \mapsto v_t)}$. Interestingly, these models can predict at an arbitrary time t in the train horizon without unrolling a sequential model up to t. Yet, as they only learn mappings from an initial condition v_0 to a function of time v_t in the train domain, they fail to predict beyond train conditions, as we show in Section 4. DINo is a new instance of spatiotemporal INR which solves this limitation via a time-continuous dynamics model of the underlying flow, $\boldsymbol{(v_t, \tau) \mapsto v_{t+\tau}}$.

B Additional Results

We present another experiments, **finer time resolution**, where we choose the inference time grid $\mathcal{T}'$ with a finer resolution than the train one $\mathcal{T}$. We consider in Table 4 a longer and 10 times finer test time grid $\mathcal{T}'$ than the train grid $\mathcal{T}$ on *Navier-Stokes*. We observe the same spatial uniform grid across *train* and *test* and perform inference on this grid. We compare DINo that performs prediction with an ODE solver, to interpolating coarser predictions obtained at the train resolution (I-DINo). We report the corresponding *test* MSE. We observe that the ODE solver accurately extrapolates outside the train temporal grid, outperforming interpolation. This confirms that DINo benefits from its continuous-time modeling of the flow, providing consistency and stability across temporal resolutions.

C Full Results

We provide in Table 5 a more detailed version of Table 2 for the space-time extrapolation problem where we report the performance *In-s* (on the observation grid) and *Out-s* (outside). We add $s = 50\%$. Then, we report in Table 6, a more detailed version of Table 3a, which includes the results of $\mathcal{X}_{\text{ts}} = \mathcal{X}_{\text{tr}}$. This corresponds to our generalization across grids problem.

D Prediction Examples

We display the test prediction of DINo (Figure 6) and I-MP-PDE (Figure 7) on *Navier-Stokes* for various subsampling levels when $\mathcal{X} = \mathcal{X}_{\text{tr}} = \mathcal{X}_{\text{ts}}$. We plot the prediction of DINo on *Wave* in the same setting in Figure 8. Predictions are performed on a 64×64 uniform grid which defines the observation grid $\mathcal{X}$ via different subsampling rates. Yellow points correspond to the observation grid $\mathcal{X}$ (*In-s*) while purple points indicate off-grid points (*Out-s*). The prediction for I-MP-PDE at $t = 0$ is the interpolated initial condition.

E Detailed Description of Datasets

We choose $\mathcal{T}$ (resp. $\mathcal{T}'$) on a regular grid in $[0, T]$ (resp. $[0, T']$) with a given temporal resolution and fix $T' = 2T$. We provide further details on the choice of these parameters and other experimental parameters, such as the number of observed trajectories.

2D Wave equation (*Wave*). It is a second-order PDE:

$$\frac{\partial^2 u}{\partial t^2} = c^2 \Delta u, \tag{10}$$

where u is a function of the displacement at each point in space w.r.t. the rest position, $c \in \mathbb{R}_+^*$ is the speed of wave traveling. We transform the equation to a first-order form, considering the input $v_t = \left(u_t, \frac{\partial u_t}{\partial t}\right)$, so that the dimension of $v_t(x)$ at each point $x \in \Omega$ is $n = 2$.

Table 5: **Space and time extrapolation.** The train and test observation grids are equal; they are subsampled with a ratio s from an uniform 64×64 grid fixed here to be the inference grid $\mathcal{X}'$. We report MSE (↓) on $\mathcal{X}'$ (on the observation grid *In-s*, outside *Out-s* or on both *Full*) and the inference time interval $\mathcal{T}'$, divided within training horizon (*In-t*, $\mathcal{T}$) and beyond (*Out-t*, outside $\mathcal{T}$) across subsampling ratios $s \in \{5\%, 25\%, 50\%, 100\%\}$. Best in **bold** and second best underlined.

	Model	Navier-Stokes				Wave			
		Train		Test		Train		Test	
		In-t	Out-t	In-t	Out-t	In-t	Out-t	In-t	Out-t
	$s = 5\%$ subsampling								
In-s	I-MP-PDE	**3.525E−5**	1.295E−3	4.554E−4	1.414E−3	**1.824E−6**	8.672E−5	1.113E−5	1.987E−4
In-s	DeepONet	4.778E−4	4.517E−3	1.060E−2	1.059E−2	2.546E−4	8.831E−3	1.501E−2	3.196E−2
In-s	SIREN	5.966E−3	1.769E−1	4.082E−2	2.150E−1	1.690E−3	1.707E−2	2.951E−2	6.955E−2
In-s	DINo	1.016E−4	**6.945E−4**	**3.623E−4**	**8.306E−4**	2.250E−6	**5.283E−6**	**7.530E−6**	**2.146E−5**
Out-s	I-MP-PDE	8.550E−3	8.515E−3	8.306E−3	8.571E−3	7.412E−4	7.414E−4	1.195E−3	1.163E−3
Out-s	DeepONet	3.475E−3	7.515E−3	1.361E−2	1.426E−2	8.624E−4	9.318E−3	1.702E−2	3.259E−2
Out-s	SIREN	8.882E−3	1.767E−1	4.314E−2	2.124E−1	2.791E−3	1.823E−2	3.359E−2	6.965E−2
Out-s	DINo	**1.076E−3**	**1.704E−3**	**1.375E−3**	**1.863E−3**	**4.285E−5**	**4.304E−5**	**6.703E−5**	**7.659E−5**
Full	I-MP-PDE	8.154E−3	8.166E−3	7.926E−3	8.225E−3	7.055E−4	7.097E−4	1.138E−3	1.116E−3
Full	DeepONet	3.330E−3	7.370E−3	1.346E−2	1.408E−2	8.331E−4	9.295E−3	1.692E−2	3.256E−2
Full	SIREN	8.741E−3	1.767E−1	4.303E−2	2.126E−1	2.738E−3	1.818E−2	3.339E−2	6.964E−2
Full	DINo	**1.029E−3**	**1.655E−3**	**1.326E−3**	**1.813E−3**	**4.088E−5**	**4.121E−5**	**6.415E−5**	**7.392E−5**
	$s = 25\%$ subsampling								
In-s	I-MP-PDE	1.447E−4	5.677E−4	**1.763E−4**	6.147E−4	**6.754E−7**	8.251E−5	**9.253E−7**	1.227E−4
In-s	DeepONet	7.500E−4	5.779E−3	9.227E−3	1.300E−2	5.196E−4	1.058E−2	1.743E−2	3.246E−2
In-s	SIREN	4.786E−3	2.178E−1	2.461E−1	3.884E−1	8.478E−4	1.282E−2	1.733E−2	5.104E−2
In-s	DINo	**8.295E−5**	**4.273E−4**	2.444E−4	**5.735E−4**	3.194E−6	**3.747E−6**	8.907E−6	**1.029E−5**
Out-s	I-MP-PDE	3.678E−4	7.748E−4	4.026E−4	8.143E−4	4.330E−5	1.200E−4	6.764E−5	1.648E−4
Out-s	DeepONet	9.503E−4	5.987E−3	9.423E−3	1.337E−2	5.891E−4	1.062E−2	1.762E−2	3.213E−2
Out-s	SIREN	5.305E−3	2.173E−1	2.428E−1	3.853E−1	9.159E−4	1.295E−2	1.798E−2	5.156E−2
Out-s	DINo	**1.081E−4**	**4.578E−4**	**2.711E−4**	**6.021E−4**	**4.192E−6**	**4.657E−6**	**1.153E−5**	**1.220E−5**
Full	I-MP-PDE	3.135E−4	7.245E−4	3.476E−4	7.658E−4	3.293E−5	1.108E−4	5.142E−5	1.545E−4
Full	DeepONet	9.016E−4	5.936E−3	9.376E−3	1.328E−2	5.722E−4	1.061E−2	1.757E−2	3.221E−2
Full	SIREN	5.180E−3	2.175E−1	2.436E−1	3.861E−1	8.995E−4	1.292E−2	1.783E−2	5.143E−2
Full	DINo	**1.020E−4**	**4.504E−4**	**2.646E−4**	**5.951E−4**	**3.949E−6**	**4.436E−6**	**1.089E−5**	**1.174E−5**
	$s = 50\%$ subsampling								
In-s	I-MP-PDE	1.153E−4	5.016E−4	**1.594E−4**	6.043E−4	**2.200E−7**	3.179E−5	**8.843E−7**	5.854E−5
In-s	DeepONet	6.214E−4	4.277E−3	5.699E−3	1.082E−2	7.581E−4	1.187E−2	1.649E−2	3.378E−2
In-s	SIREN	4.911E−3	6.815E−1	1.607E−1	6.889E−1	5.134E−4	1.481E−2	3.086E−2	8.196E−2
In-s	DINo	**8.151E−5**	**2.920E−4**	2.004E−4	**4.283E−4**	3.277E−6	**3.659E−6**	8.978E−6	**9.572E−6**
Out-s	I-MP-PDE	1.186E−4	5.010E−4	**1.626E−4**	6.132E−4	**9.638E−7**	3.153E−5	**2.367E−6**	5.574E−5
Out-s	DeepONet	6.851E−4	4.343E−3	5.740E−3	1.099E−2	7.842E−4	1.185E−2	1.679E−2	3.391E−2
Out-s	SIREN	5.067E−3	6.867E−1	1.599E−1	6.845E−1	5.354E−4	1.492E−2	3.113E−2	8.333E−2
Out-s	DINo	**9.175E−5**	**3.041E−4**	2.116E−4	**4.409E−4**	3.277E−6	**3.659E−6**	8.978E−6	**9.572E−6**
Full	I-MP-PDE	1.170E−4	5.013E−4	**1.611E−4**	6.088E−4	**6.021E−7**	3.166E−5	**1.646E−6**	5.710E−5
Full	DeepONet	6.541E−4	4.311E−3	5.720E−3	1.091E−2	7.715E−4	1.186E−2	1.665E−2	3.385E−2
Full	SIREN	4.995E−3	6.841E−1	1.603E−1	6.867E−1	5.246E−4	1.486E−2	3.100E−2	8.265E−2
Full	DINo	**8.677E−5**	**2.982E−4**	2.062E−4	**4.348E−4**	3.380E−6	**3.751E−6**	9.251E−6	**9.710E−6**
	$s = 100\%$ subsampling								
Full	CNODE	2.319E−2	9.652E−2	2.305E−2	1.143E−1	2.337E−5	5.280E−4	3.057E−5	7.288E−4
Full	MP-PDE	1.140E−4	5.500E−4	**1.785E−4**	5.856E−4	**1.718E−7**	1.993E−5	**9.256E−7**	4.261E−5
Full	MNO	**3.190E−5**	8.678E−4	2.763E−4	8.946E−4	9.381E−6	4.890E−3	1.993E−4	6.128E−3
Full	DeepONet	1.375E−3	6.573E−3	9.704E−3	1.244E−2	6.431E−4	1.293E−2	1.847E−2	3.317E−2
Full	SIREN	1.066E−3	4.336E−1	3.874E−1	1.037E0	3.674E−4	9.956E−3	3.013E−2	7.842E−2
Full	MFN	1.651E−3	1.037E0	2.106E−1	1.059E0	1.408E−4	1.763E−1	4.735E−3	2.274E−1
Full	DINo (no sep.)	3.235E−4	1.593E−3	7.850E−4	1.889E−3	2.641E−6	4.081E−5	5.977E−5	2.979E−4
Full	DINo	8.339E−5	**3.115E−4**	2.092E−4	**4.311E−4**	3.309E−6	**3.506E−6**	9.495E−6	**9.946E−6**

Table 6: **Generalization across grids.** $\mathcal{X}_{\text{tr}}, \mathcal{X}_{\text{ts}}$ are subsampled with different ratios $s_{\text{tr}} \neq s_{\text{ts}} \in \{5, 50, 100\}\%$ from the same uniform 64×64 grid. We report *test* MSE within $\mathcal{X}_{ts}$ (*In-s*). **Best** in bold.

		$\mathcal{X}_{\text{ts}} = \mathcal{X}_{\text{tr}}$		$\mathcal{X}_{\text{ts}} \neq \mathcal{X}_{\text{tr}}$					
Subsampling	Test →	$s_{\text{ts}} = s_{\text{tr}}$		$s_{\text{ts}} = 5\%$		$s_{\text{ts}} = 50\%$		$s_{\text{ts}} = 100\%$	
Train ↓		In-t	Out-t	In-t	Out-t	In-t	Out-t	In-t	Out-t
$s_{\text{tr}} = 5\%$	MP-PDE	**1.967E−4**	**6.631E−4**	1.330E−1	3.852E−1	1.859E−1	6.680E−1	2.105E−1	7.120E−1
	DINo	3.623E−4	8.306E−4	**1.494E−3**	**2.291E−3**	**1.257E−3**	**1.883E−3**	**1.287E−3**	**1.947E−3**
$s_{\text{tr}} = 50\%$	MP-PDE	**1.346E−4**	5.110E−4	4.494E−2	9.403E−2	4.793E−3	1.997E−2	6.330E−3	3.712E−2
	DINo	2.004E−4	**4.283E−4**	**2.470E−4**	**4.697E−4**	**2.073E−4**	**4.284E−4**	**2.058E−4**	**4.361E−4**
$s_{\text{tr}} = 100\%$	MP-PDE	**1.785E−4**	5.856E−4	1.358E−1	3.355E−1	1.182E−2	2.664E−2	**1.785E−4**	5.856E−4
	DINo	2.092E−4	**4.311E−4**	**2.495E−4**	**4.805E−4**	**2.109E−4**	**4.325E−4**	2.092E−4	**4.311E−4**

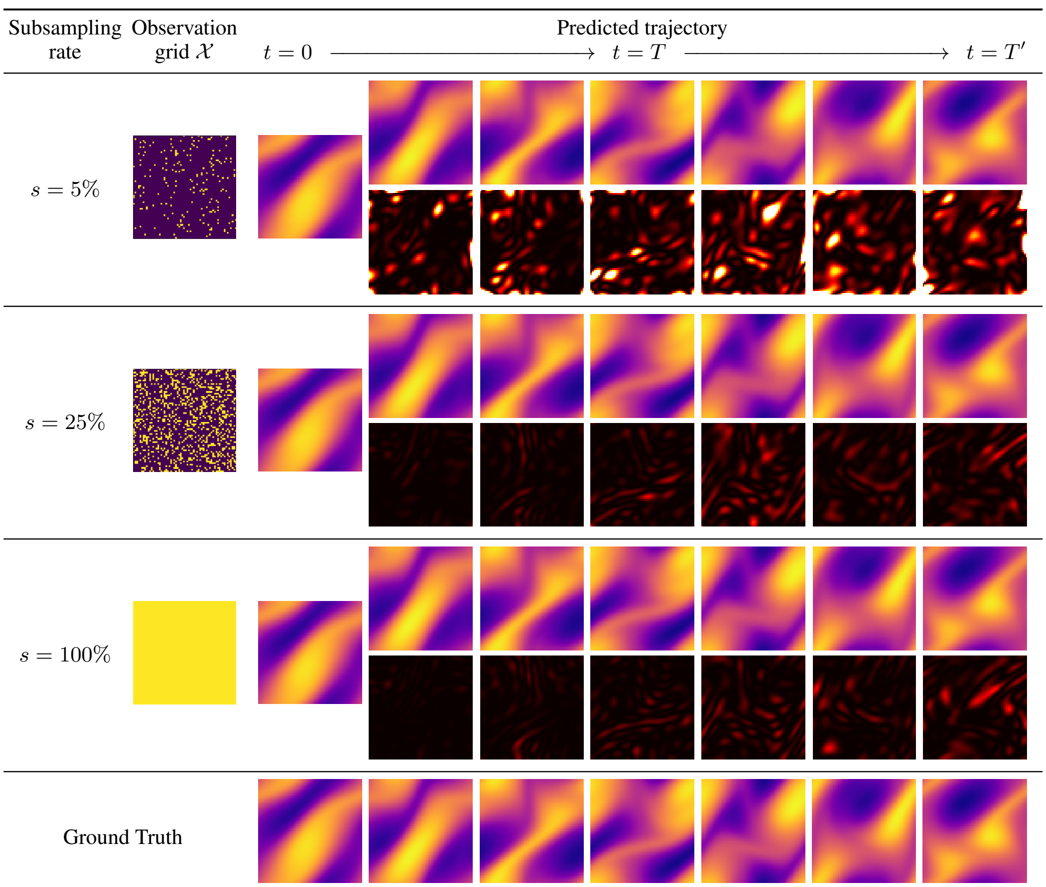


Figure 6: Prediction MSE per frame for **DINo** on *Navier-Stokes* with its corresponding observed train and test grid $\mathcal{X}$. For each model, the first row contains the predicted trajectory from 0 to T', the second row is the corresponding error maps w.r.t. the reference data (the darker the pixel, the lower the error).

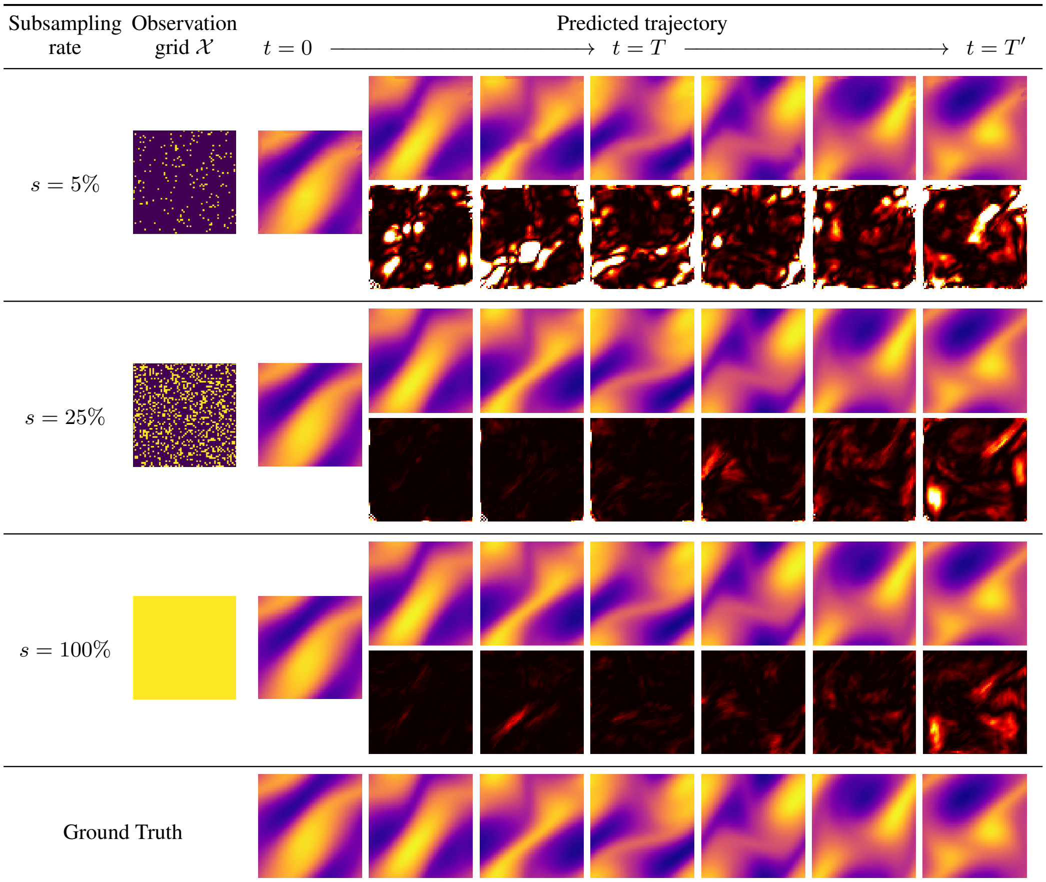


Figure 7: Prediction MSE per frame for **I-MP-PDE** on *Navier-Stokes* with its corresponding observed train and test grid $\mathcal{X}$. For each model, the first row contains the predicted trajectory from 0 to T', the second row is the corresponding error maps w.r.t. the reference data (the darker, the lower the error).

We generate our dataset for speed $c = 2$ with periodic boundary condition. The domain is $\Omega = [-1, 1]^2$. For initial conditions $v_0 = \left(u_0, \left.\frac{\partial u_t}{\partial t}\right|_{t=0}\right)$, the initial displacement u_0 is Gaussian:

$$u_0(x; a, b, r) = a \exp\left(-\frac{(x-b)^2}{2r^2}\right), \tag{11}$$

where the height of the peak displacement is $a \sim \mathcal{U}(2, 4)$, the location of the peak displacement is $(b_1, b_2) \sim \mathcal{U}(-1, 1)$, and the standard deviation is $r \sim \mathcal{U}(0.25, 0.3)$. The initial time derivative is $\left.\frac{\partial u_t}{\partial t}\right|_{t=0} = 0$. Each snapshot is generated on a uniform grid of 64×64. Each sequence is generated with fixed interval $\delta t = 0.25$. We set the training horizon $T = 2.25$ and the inference horizon $T = 4.75$. We generated 512 training trajectories and 32 test trajectories.

2D Navier Stokes (*Navier-Stokes*, [43]). This dataset describes an incompressible fluid dynamics by:

$$\frac{\partial w}{\partial t} = -u\nabla w + \nu\Delta w + f, \quad w = \nabla \times u, \quad \nabla u = 0, \tag{12}$$

where u is the velocity field and w the vorticity. u, w lie on a spatial domain with periodic boundary conditions, ν is the viscosity and f is a constant forcing term. The input v_t is w_t $(n = 1)$. ν is the viscosity and f is the constant forcing term in the domain Ω.

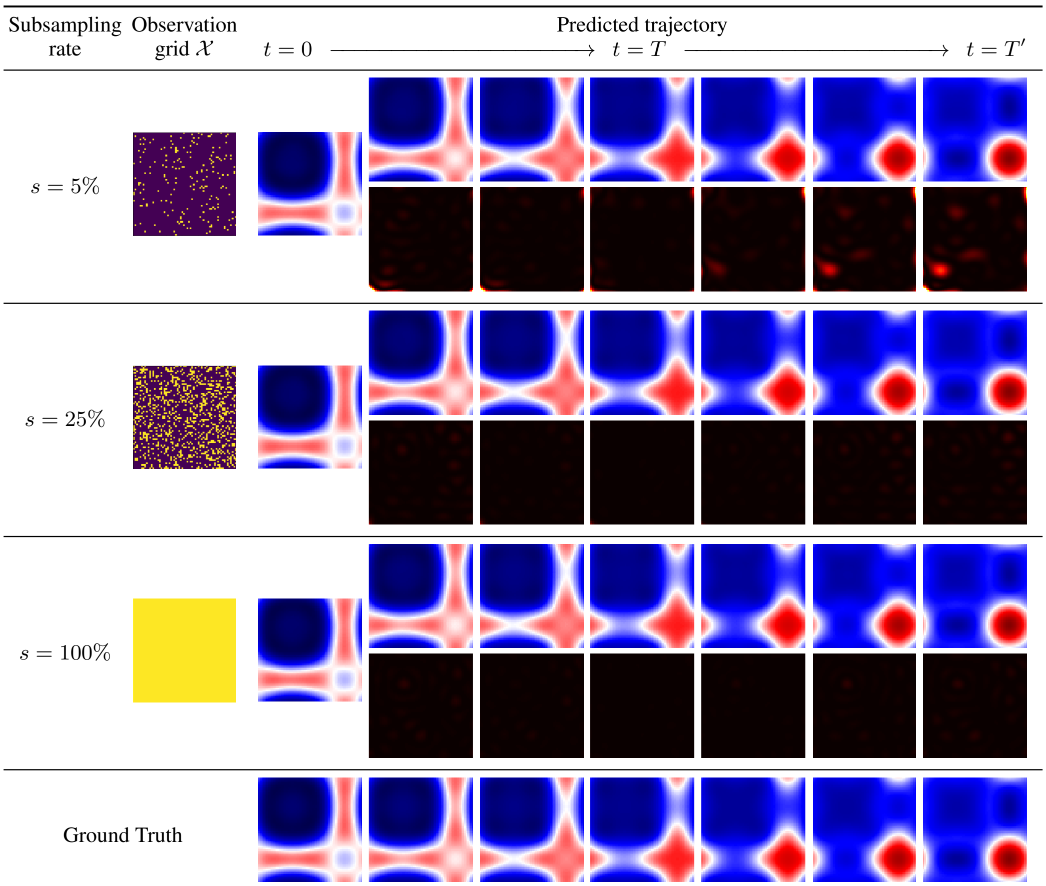


Figure 8: Prediction MSE per frame for **DINo** on *Wave* with its corresponding observed train and test grid $\mathcal{X}$. For each model, the first row contains the predicted trajectory from 0 to T', the second row is the corresponding error maps w.r.t. the reference data (the darker the pixel, the lower the error).

The spatial domain is $\Omega = [-1, 1]^2$, the viscosity is $\nu = 1 \times 10^{-3}$, the forcing term is set as:

$$\forall x \in \Omega, f(x_1, x_2) = 0.1\Big(\sin\big(2\pi(x_1 + x_2)\big) + \cos\big(2\pi(x_1 + x_2)\big)\Big). \tag{13}$$

The full spatial grid is of dimension 64×64 or 256×256 according to experiments in Section 4. We sample initial conditions as in Li et al. [28] to create different trajectories. The first 20 steps of the trajectories are cut off as they are too noisy and not informative in terms of dynamics. Trajectories are collected with $\delta t = 1$. We set the training horizon $T = 19$ and the inference horizon $T' = 39$. We generated 512 training trajectories and 32 test trajectories.

3D spherical shallow water (*Shallow-Water*, [16]). The following problem is originally presented for testing numerical models of global shallow-water equations. The shallow water equations is written as:

$$\begin{aligned} \frac{\mathrm{d}u}{\mathrm{d}t} &= -fk \times u - g\nabla h + \nu\Delta u, \\ \frac{\mathrm{d}h}{\mathrm{d}t} &= -h\nabla \cdot u + \nu\Delta h. \end{aligned} \tag{14}$$

where $\frac{\mathrm{d}}{\mathrm{d}t}$ is the material derivative, k is the unit vector orthogonal to the spherical surface, u is the velocity field tangent to the surface of the sphere, which can be transformed into the vorticity $w = \nabla \times u$, h is the thickness of the sphere. Note that the data we observe at each time t is $v_t = (w_t, h_t)$. f, g, ν, Ω are parameters of the Earth (cf. [16] for details).

Algorithm 1: DINo pseudo-code

Training: **Input :** $\mathcal{D} = \{v_\mathcal{T}\}$, $\{\alpha_\mathcal{T}^v\}_{v\in\mathcal{D}} \leftarrow \{0\}$, $\phi \leftarrow \phi_0$, $\psi \leftarrow \psi_0$;
while *not converged* **do**
  for $v \in \mathcal{D}$ **do** $\alpha_\mathcal{T}^v \leftarrow \alpha_\mathcal{T}^v - \eta_\alpha \nabla_{\alpha_\mathcal{T}^v} \ell_{\text{dec}}(\phi, \alpha_\mathcal{T}^v)$; /* Modulation */
  $\phi \leftarrow \phi - \eta_\phi \nabla_\phi \Big(\sum_{v\in\mathcal{D}} \ell_{\text{dec}}(\phi, \alpha_\mathcal{T}^v)\Big)$; /* Hypernetwork */
  $\psi \leftarrow \psi - \eta_\psi \nabla_\psi \Big(\sum_{v\in\mathcal{D}} \ell_{\text{dyn}}(\psi, \alpha_\mathcal{T}^v)\Big)$; /* Dynamics */
Test: **Input :** $\mathcal{D}'_0 = \{v_0\}$, $\{\alpha_0^v\}_{v\in\mathcal{D}'} \leftarrow \{0\}$, $\phi^\star$, $\psi^\star$, $\mathcal{T}' \neq \mathcal{T}$;
while *not converged* **do**
  for $v \in \mathcal{D}'$ **do** $\alpha_0^v \leftarrow \alpha_0^v - \eta \nabla_{\alpha_0^v} \ell_{\text{dec}}(\phi^\star, \alpha_0^v)$; /* Modulation */
for $v \in \mathcal{D}', t \in \mathcal{T}'$ **do** $\alpha_t^v \leftarrow \alpha_0^v + \int_0^t f_{\psi^\star}(\alpha_\tau^v)\,\mathrm{d}\tau$; /* Unroll dynamics */

The initial conditions are slightly modified from [16], detailed below, to create symmetric phenomena on the northern and southern hemisphere. The initial zonal velocity u_0 contains two non-null symmetric bands in the both hemispheres, which are parallel to the circles of latitude. At each latitude and longitude $\phi, \theta \in [-\pi/2, \pi/2] \times [-\pi, \pi]$:

$$u_0(\phi,\theta) = \begin{cases} \left(\dfrac{u_{\max}}{e_{\mathrm{n}}} \exp\left(\dfrac{1}{(\phi-\phi_0)(\phi-\phi_1)}\right), 0\right) & \text{if } \phi \in (\phi_0, \phi_1), \\ \left(\dfrac{u_{\max}}{e_{\mathrm{n}}} \exp\left(\dfrac{1}{(\phi+\phi_0)(\phi+\phi_1)}\right), 0\right) & \text{if } \phi \in (-\phi_1, -\phi_0), \\ (0,0) & \text{otherwise.} \end{cases} \tag{15}$$

where $u_{\max}$ is the maximum velocity, $\phi_0 = \pi/7, \phi_1 = \pi/2 - \phi_0$, and $e_{\mathrm{n}} = \exp\left(-4/(\phi_1-\phi_0)^2\right)$. The water height h_0 is initialized by solving a boundary value condition problem as in Galewsky et al. [16]. It is then perturbed by adding the following h'_0 to h_0:

$$h'_0(\phi,\theta) = \hat{h}\cos(\phi)\exp\left(-\left(\frac{\theta}{\alpha}\right)^2\right)\left[\exp\left(-\left(\frac{\phi_2-\phi}{\beta}\right)^2\right) + \exp\left(-\left(\frac{\phi_2+\phi}{\beta}\right)^2\right)\right]. \tag{16}$$

where $\phi_2 = \pi/4$, $\hat{h} = 120\,\mathrm{m}, \alpha = 1/3, \beta = 1/15$ are constants defined in [16].

We simulate this phenomenon with Dedalus [6] on a latitude-longitude (lat-lon) grid. The size of the grid is 128 (lat)×256 (lon). We take different initial conditions by sampling $u_{\max} \sim \mathcal{U}(60, 80)$ to generate long trajectories. These long trajectories are then sliced into shorter ones. For simulation, we take one snapshot per hour (of internal simulation time), i.e. $\delta t = 1\,\mathrm{h}$. We stop the simulation at the 320$^{\text{th}}$ hour. To construct a dataset rich of dynamical phenomena, we take the snapshots within the last 160 h in a long trajectory and slice them into 8 shorter trajectoires. Also note that the data is scaled into a reasonable range: the height h is scaled by a factor of 3×10^3, and the vorticity w by a factor 2. In each short trajectory, $T = 9\,\mathrm{h}$ and $T' = 19\,\mathrm{h}$. In total, we generated 16 long trajectories (i.e. 128 short trajectories) for train, 2 for test (i.e. 16 short trajectories).

F Implementation

Algorithm. We detail the algorithm of DINo for training and test via pseudo-code in Algorithm 1.

Implementation details. We use PyTorch [35] to implement DINo and the baselines. The dynamics model f_ψ is a multilayer perceptron. Its input and output size are same as the size of latent space d_α. All hidden layers share the same size. DINo's parameters are initialized with the default initialization in PyTorch. The frequency parameters in FourierNet are scaled by a factor, considered as a hyperparameter. For dynamics learning, we use an RK4 integrator and apply exponential Scheduled Sampling [2] to stabilize training. ω is fixed at initialization to reduce the number of optimized parameters without loss of performance. In practice, modulations of α_t are learned channel-wise such

Table 7: DINO's hyperparameters.

Hyperparameter	Navier-Stokes	Wave	Shallow-Water
Decoder $D_\phi = I_{h_\phi}$			
Number of layers	3	3	6
Number hidden channels	64	64	256
Frequency scale factor	64	64	64
Size of latent space d_α	100	50	300
Dynamics model f_ψ			
Number of layers	4	4	4
Hidden layer size	512	512	800
Activation function	Swish	Swish	Swish
Optimization			
Learning rate η_ϕ	10^{-2}	10^{-2}	10^{-2}
Learning rate η_α	10^{-3}	10^{-3}	10^{-3}
Learning rate η_ψ	10^{-3}	10^{-3}	10^{-3}
Number of epochs	12 000	12 000	12 000
Batch size i.e. sequences per batch	64	64	16

that $I_\theta : \Omega \to \mathbb{R}^{d_c}$ has separate parameters per output dimension to make predictions less correlated across channels. We optimize all parameters using Adam [21] with $(\beta_1, \beta_2) = (0.9, 0.999)$.

Hyperparameters. We list the hyperparameters of DINO for each dataset in Table 7. In practice, we observe it is beneficial to decay the learning rates η_ϕ, η_α when the loss reaches a plateau.

Baselines implementation. We detail in the following the hyperparameters and architectures used in our experiments for the considered baselines, which we reimplemented for our paper.

- CNODE is implemented with four 2D convolutional layers with 64 hidden features, ReLU activations, 3×3 kernel and zero padding. Learning rate is fixed to 10^{-3}. We use an adjoint method for integration like [9].
- MNO: we use the FNO architecture in [28] with three FNO blocks, GeLU activations, 12 modes and a width of 32. Learning rate is fixed to 10^{-3}.
- DeepONet: we consider an autoregressive formulation of DeepONet. We choose a width of 1000 for hidden features with a depth of 4 for both trunk and branch nets with ReLU activations. Learning rate is fixed to 10^{-5}.
- MP-PDE: we adapt the implementation in [4] to handle 2D and 3D PDEs. We use a time window of 1 with pushforward trick. Batch size and number of neighbors are fixed to 8. Learning rate is fixed to 10^{-3}. We use ReLU activations.
- SIREN: to represent data in space and time, SIREN takes space and time coordinates (x, t) as input. To handle multiple trajectories, we concatenate an optimizable per-trajectory context code α to the coordinates like in DINO. We fix the hidden layer size of SIREN to 256. We initialize the parameters and use the default input scale as in [41]. The size of the context code is $d_\alpha = 800$. The learning rate is 10^{-3}.
- MFN: similarly to the previous SIREN baseline, we concatenate the per-trajectory context code to space and time coordinates at the first layer. The hidden layer size is fixed to 256 and we use the default parameter initialization with a frequency scale of $\times 64$ as DINO. The size of the context code is $d_\alpha = 800$. The learning rate is 10^{-3}.