POLITECNICO DI TORINO
Repository ISTITUZIONALE

A Comparative Study of Neural Ordinary Differential Equations and Neural Operators for Modeling Temporal Dynamics

Original
A Comparative Study of Neural Ordinary Differential Equations and Neural Operators for Modeling Temporal Dynamics / Celia, M., Monaco, S., Apiletti, D.. - In: NEURAL COMPUTING & APPLICATIONS. - ISSN 1433-3058. - ELETTRONICO. - 37:30(2025), pp. 25319-25338. [10.1007/s00521-025-11580-0]

Availability:
This version is available at: 11583/3002528 since: 2025-09-09T09:22:02Z

Publisher:
Springer

Published
DOI:10.1007/s00521-025-11580-0

Terms of use:

This article is made available under terms and conditions as specified in the corresponding bibliographic description in the repository

Publisher copyright

(Article begins on next page)

28 November 2026

ORIGINAL ARTICLE

A comparative study of neural ordinary differential equations and neural operators for modeling temporal dynamics

Matteo Celia[1] · Simone Monaco[1] · Daniele Apiletti[1]

Received: 17 March 2025 / Accepted: 14 August 2025 / Published online: 9 September 2025
© The Author(s) 2025

Abstract
Capturing the dynamics of relational systems is a key challenge in the natural sciences, with applications ranging from simulating molecular interactions to analyzing particle mechanics. Machine learning approaches have made significant progress in this area by using graph neural networks to learn and visualize spatial interactions effectively. Neural ordinary differential equations (Neural ODEs) and neural operators (NO) represent two distinct paradigms. However, a clear comparative understanding of when to prefer one over the other is still lacking. To address this gap, we present the first systematic comparison between two representative architectures: EGNO (Equivariant Graph Neural Operator) and SEGNO (Second-order Equivariant Graph Neural Ordinary Differential Equation). Through a series of experiments, we investigate their strengths and limitations in various simulation scenarios in the multi-step trajectory prediction tasks. Specifically, we employ rollout strategies and different input/output configurations, including multiple and irregularly sampled time steps. Our findings highlight a key trade-off between precision and stability that is central to model selection. SEGNO demonstrates superior robustness and stability over long prediction horizons, making it well-suited for tasks requiring reliable long-term forecasting. Conversely, EGNO offers higher precision during early stages of the trajectory and better leverages diverse training configurations, thanks to its discretization-invariant design. In summary, Neural Operators (EGNO) are preferable when short-term accuracy and data efficiency are critical, while Neural ODEs (SEGNO) are advantageous for scenarios demanding stable long-term predictions. This work not only clarifies the practical advantages of each approach but also lays the groundwork for informed model selection and future hybrid strategies in dynamical system modeling.

Keywords Spatio-temporal graph dynamics · Particle systems forecasting · Neural operator · Neural ordinary differential equation

1 Introduction

Modeling n-body systems, such as those found in molecular dynamics [1], astrophysics [2], and fluid simulations [3], remains a fundamental challenge in science and engineering. These systems aim to predict the behavior of multiple interacting entities governed by forces such as gravity, electromagnetism, or intermolecular potentials. Although traditional methods have achieved remarkable success in addressing these problems [4], they face

Matteo Celia and Simone Monaco contributed equally to this work.

This work not only clarifies the practical advantages of each approach but also lays the groundwork for informed model selection and future hybrid strategies in dynamical system modeling.

notable challenges, especially when applied to large and complex systems. Recently, data-driven approaches utilizing machine learning (ML), including deep learning (DL) techniques, have emerged as promising alternatives or complements to these traditional methods [5–7]. The reason is the fact that traditional numerical methods, such as molecular dynamics (MD), generally require a step-by-step solution of the equation of motion governing the systems. This involves calculating the forces between every pair of particles, which scales poorly with the number of particles, typically $O(N^2)$, although clever algorithms can reduce this. Moreover, MD simulations often require extremely small time steps to ensure accuracy and stability, making long simulations computationally expensive. Similar challenges arise in astrophysics, where simulating galaxy evolution requires tracking gravitational interactions among billions of stars over millions of years.

These traditional methods, while powerful, have clear limitations in terms of computational demands, which increase dramatically with the number of bodies and the required resolution in time and space. Additionally, modeling these systems often relies on detailed parameterizations of forces (e.g., Lennard–Jones potentials in MD or gravitational potentials in astrophysics), either approximations or empirically derived. Such parameterizations may not generalize well to new systems. In addition, these simulations struggle to bridge vastly different scales, such as atomic interactions and macroscopic behavior.

In this context, ML algorithms can significantly accelerate the process by learning patterns directly from data, eliminating the need for explicit force calculations or iterative time stepping [8, 9]. Neural networks, for example, can approximate complex non-linear relationships, making them particularly effective in predicting the dynamics of n-body systems. Once trained, these models deliver predictions far faster than traditional methods while also exhibiting generalization capabilities to unseen systems.

This approach is gaining popularity as advances in computational simulations and experimental techniques have produced extensive datasets for training models. These datasets span diverse fields, including scientific problems governed by ordinary or partial differential equations [10–12], molecular dynamics [13, 14], and astrophysics [15, 16], further enabling the application of deep learning to complex n-body problems.

Unlike traditional methods, which are grounded in well-understood physical principles, machine learning models often operate as black boxes, making their predictions less interpretable. This lack of transparency can result in violations of conservation laws (e.g., energy, momentum, or symmetries), leading to unphysical or inconsistent outcomes. A promising direction to address this limitation involves incorporating prior knowledge into training strategies through invariances, soft constraints, or architectural biases [17, 18].

In the context of simulating particle systems, Graph Neural Networks (GNNs) naturally embed the well-known inductive biases. These models inherently handle the permutation invariance of particles and are designed to be independent of the number of particles and their relative neighbors, making them highly adaptable to such tasks [9]. Taking this a step further, an essential inductive bias for these problems is the incorporation of translational and rotational invariances, which is achieved by Equivariant Graph Neural Networks (EGNNs).

For mapping the temporal evolution of target systems, GNNs have traditionally been used to predict states at specific time steps, employing discrete state transformation layers to learn direct mappings between adjacent states. However, this approach often neglects the continuous nature of system trajectories. Addressing this limitation involves incorporating inductive biases that enable neural networks to capture the underlying dynamics.

An approach models the dynamics as an Ordinary Differential Equation (ODE), as seen in the neural ODE framework [19]. Another treats the network as an operator that maps infinite-dimensional functions, representing the system's non-discretized states across different points of the trajectory, as in the NO framework [20].

Two notable EGNN architectures exemplify these directions are SEGNO (Second-order Equivariant Graph Neural Ordinary Differential Equation) [21], which follows the neural ODE paradigm, and EGNO (Equivariant Graph Neural Operator) [22], which adopts the NO one.

Both approaches show promise for addressing the challenges under analysis, but it remains uncertain which framework is superior overall or in specific aspects, particularly when modeling the temporal evolution of complex systems like charged particle systems.

This study performed a systematic comparison of the SEGNO and EGNO models to uncover their respective strengths and weaknesses. Using data on their performance within a specific application domain, the research seeks to offer valuable insights into the suitability of each approach. We evaluated both models on a benchmark particle system and designed different custom training strategies to enhance both models, supporting diverse input/output designs, including multiple and irregularly sampled time steps. The final goal is to show which model best suits each applicative scenario.

The paper is organized as follows. Section 2 recalls the most influential works in the literature related to the scope of the paper. Section 3 introduces the analyzed models and the adopted techniques, along with the proposed experimental design. Finally, Section 4 collects the results, and Section 5 discusses them and draws the conclusions.

2 Related works

2.1 Graph neural networks

GNN [23] is a family of deep learning models designed to handle graph-structured data. They address tasks related to graphs, such as particle systems, in an end-to-end manner. These architectures are based on the message passing framework [24], on top of which different architectures have been developed over the years to deal with spectral representations [25], include the attention mechanism [26], and solve common problems such as over-smoothing [27].

GNNs can be adapted to address various graph-related tasks, including node classification, graph classification, link prediction, and graph generation. However, traditional GNNs struggle to effectively deal with spatial and temporal relationships among nodes and graphs [28].

Particle systems are spatio-temporal graphs, where node attributes evolve over time. To handle complex structural and temporal information, considerable research attention has been devoted to dynamic graph neural networks (DyGNNs) [29, 30]. A classic of DyGNNs first adopts a GNN to aggregate structural information for graph at each time, followed by a sequence model like RNN [31–33] or temporal self-attention [34] to process temporal information. To obtain more fine-grained continuous node embeddings in dynamic graphs, some work further leverages neural interaction processes [35] and ordinary differential equations [36].

Particle systems can be modeled as spatio-temporal graphs, where node attributes evolve over time. To handle their complex structural and temporal dynamics, significant research has focused on dynamic graph neural networks (DyGNNs) [29, 30]. A common approach in DyGNNs involves the use of a GNN to aggregate structural information at each time step, followed by recurrent models to handle the temporal evolution [31–33]. These methods are constrained by their reliance on time discretization for functionality. To address this limitation, other approaches have emerged, focusing on finer continuous node embeddings in dynamic graphs. These include using neural interaction processes [35], incorporating neural ODEs [36], or NOs [37].

2.2 Equivariant graph neural networks

Traditional GNNs have limitations when dealing with physical systems, mainly due to their inability to naturally incorporate crucial physical inductive biases and to their inability to learn on positional graphs.

Various forms and methods have been proposed to achieve $E(3)$ or $SE(3)$ equivariance. Thomas et al. [38] and Fuchs et al. [39] utilize the spherical harmonics to compute the basis for the transformations. A downside of this method is that the spherical harmonics need to be recomputed, which can be expensive. Currently, an extension of this method to arbitrary dimensions is unknown. Finzi et al. [40] parametrize transformations by mapping kernels in the Lie Algebra. However, this method makes the neural network output stochastic in certain situations, which may be undesirable. Horie et al. [41] propose a set of isometric invariant and equivariant transformations for

Graph Neural Networks. Köhler et al. [42, 43] propose an $E(n)$ equivariant network to model 3D point clouds, but the method is only defined for positional data on the nodes without any feature dimensions.

Equivariant GNNs [44, 45] were developed to address some of these limitations by explicitly embedding geometric symmetries into their architecture. They are designed to ensure that their outputs transform in predictable ways when the inputs undergo certain transformations. By enforcing this equivariance, these models align naturally with the structure of physical laws, reducing the need for the model to "learn" these symmetries from the data and ensuring that the predictions respect these kinds of fundamental principles [46]. EGNNs possess equivariance to roto-translational transformations in Euclidean space, which has been demonstrated as a vital inductive bias to improve generalization [23, 39, 42, 47].

2.3 Neural ordinary differential equation

Neural ODEs [19] represent a class of machine learning models designed to capture continuous-time dynamics. Recent advances have combined neural ODEs with principles from Hamiltonian mechanics to model interactions in complex systems [48, 49]. These approaches have been further enhanced by integrating Graph Neural Networks (GNNs). For example, GDE [50] and HOGN [51] couple GNNs with differentiable ODE solvers, while GNODE [52] introduces graph-based Neural ODEs with Newton's third law as inductive bias. Moreover, second-order Graph Neural ODEs have demonstrated the ability to minimize position discrepancies, ensuring bounded errors in instantaneous acceleration and position.

Notably, Graph Network-based Simulators (GNS) [7] leverage acceleration optimization but focus on learning average accelerations derived from observed trajectories. In contrast, SEGNO [21] advances this approach by learning instantaneous accelerations through a second-order framework that incorporates equivariant networks.

2.4 Neural operators

Neural operators [20] are machine learning frameworks designed to learn mappings between Banach spaces, which are continuous function spaces. These models have been extensively utilized as data-driven surrogates for solving partial differential equations (PDEs) and ordinary differential equations (ODEs) in various scientific contexts.

Among these, the Fourier Neural Operator (FNO) [53] has established itself as a leading method for addressing PDE problems across diverse scientific fields.

While prior research primarily focused on functions defined over spatial domains [54, 55] or temporal functions with simple scalar outputs [56], a recent development by Xu et al. [22] introduced EGNO. This model integrates the strengths of equivariant architectures to handle geometric dynamics represented as temporal functions while maintaining the symmetry of Euclidean space.

3 Materials and methods

3.1 Problem formulation

We can define a spatial-temporal graph representing the particle system evolution as $\mathcal{G}^{(t)} = (\mathbf{V}, \mathbf{E}^{(t)}, \mathbf{X}^{(t)}, \mathbf{W}^{(t)})$, indexed by time $t \in [0, T]$, where T is the trajectory length, $\mathbf{V}$ is the set of vertices or nodes, $\mathbf{E}^{(t)}$ is the set of edges and $\mathbf{X}^{(t)}$ is the node feature matrix. As the particle system evolves over time, both the edge connections and their associated features are expected to change, implying a time-varying graph topology. Suppose that we have a system of N particles, $X^{(t)}$ at each timestep will be represented as the feature map $[\mathbf{h}, \mathbf{Z}^{(t)}]$, where $\mathbf{h} = (\mathrm{h}_1, \ldots, \mathrm{h}_N) \in \mathbb{R}^{N \times k}$ is the invariant and static node feature matrix of dimension k (e.g., atom types and charges),

and $\mathbf{Z}^{(t)} \in \mathbb{R}^{N\times m\times 3}$ is the matrix composed of m directional 3D tensors, which represent the equivariant features. Finally, features $\mathbf{W}^{(t)}$ are typically derived from the dynamic node states, such as inter-particle distances or potential energies.

Within this portrait, we are going to present the SEGNO and EGNO architectures, emphasizing their design concerning this input data structure.

3.2 SEGNO

Differently from previous models that use equivariant GNNs to fit discrete kinematic states, SEGNO [21] introduces Neural ODEs to approximate a continuous trajectory between two observed states. Furthermore, to better estimate the underlying dynamics, SEGNO is built upon second-order motion equations to the position and velocity of physical systems. Theoretically, the uniqueness of the learned latent trajectory of SEGNO has been proven, and further provided an upper bound on the discrepancy between the learned and the actual latent trajectory.

SEGNO combines Neural ODEs with equivariant GNNs to model the continuous latent trajectory of dynamical systems. Starting from initial states (positions and velocities), SEGNO computes future positions by integrating velocities, which are derived by integrating accelerations. Accelerations are parameterized through an Equivariant GNN, leveraging the system's trajectory and node features as input. To approximate the latent trajectory, SEGNO employs numerical integrators, like the Euler method, dividing the time interval into smaller steps and iteratively updating positions and velocities. This design, which is common to Neural ODEs architectures, allows SEGNO to reuse GNN building blocks, reducing the overall number of parameters.

In a dynamic systems, the ODE formulation is employed to model the latent continuous trajectory with initial system states defined by the set of positions and velocities $(\boldsymbol{q}^{(t_0)}, \dot{\boldsymbol{q}}^{(t_0)})$. The position $\boldsymbol{q}^{(t_0+t')}$ for any $t' \in [0, T]$ is calculated as

$$\begin{aligned}\phi_{t',g}(\boldsymbol{q}^{(t_0)}) := \boldsymbol{q}^{(t_0+t')} &= \boldsymbol{q}^{(t_0)} + \int_{t_0}^{t_0+t'} \left(\dot{\boldsymbol{q}}^{(t_0)} + \int_{t_0}^{t} f(\boldsymbol{q}^{(m)}, \boldsymbol{h})\, dm\right) dt \\ &= \boldsymbol{q}^{(t_0)} + \int_{t_0}^{t_0+t'} g(\boldsymbol{q}^{(t)}, \boldsymbol{h})\, dt,\end{aligned} \tag{1}$$

with g being a mapping from the trajectory to its first-order derivative $\dot{\boldsymbol{q}}^{(t)}$.

Since most physical dynamical systems follow Newton's Second Law of Motion, relating forces and accelerations, their natural formulation is in terms of second-order differential equations. Then, their accurate modeling also implies capturing the evolution of velocity over time. Therefore, we extend the modeling framework to explicitly recover the velocity trajectory:

$$\psi_{t',g,f}(\boldsymbol{q}^{(t_0)}) := \dot{\boldsymbol{q}}^{(t_0+t')} = g(\boldsymbol{q}^{(t_0)}, \boldsymbol{h}) + \int_{t_0}^{t_0+t'} f(\boldsymbol{q}^{(t)}, \boldsymbol{h})\, dt. \tag{2}$$

The second-order inductive bias is then obtained parametrizing the acceleration function with

$$\ddot{\boldsymbol{q}}_\theta^{(t)} = f_\theta(\boldsymbol{q}^{(t)}, \boldsymbol{h}). \tag{3}$$

Here f_θ, which approximates f, is an equivariant GNN. Having $\boldsymbol{q}_\theta$ the approximated trajectory generated by f_θ and satisfying the initial conditions $\boldsymbol{q}_\theta^{(t_0)} = \boldsymbol{q}^{(t_0)}, \dot{\boldsymbol{q}}_\theta^{(t_0)} = \dot{\boldsymbol{q}}^{(t_0)}$ and following Eq. 1, the predicted position of SEGNO at time $t_1 = t_0 + \Delta t$ can be represented by

$$\begin{aligned}\phi_{T,g_\theta}(\boldsymbol{q}^{(t_0)}) = \boldsymbol{q}_\theta^{(t_1)} &= \boldsymbol{q}^{(t_0)} + \int_{t_0}^{t_0+\Delta t} g_\theta(\boldsymbol{q}^{(t)}, \boldsymbol{h})\, dt \\ &= \boldsymbol{q}_\theta^{(t_0)} + \int_{t_0}^{t_0+\Delta t} \left(\boldsymbol{q}_\theta^{(t_0)} + \int_{t_0}^{t} f_\theta(\boldsymbol{q}^{(m)}, \boldsymbol{h})\, dm\right) dt \\ &= \boldsymbol{q}_\theta^{(t_0)} + \int_{t_0}^{t_0+\Delta t} \psi_{t,g_\theta,f_\theta}(\boldsymbol{q}_\theta^{(t_0)})\, dt,\end{aligned} \tag{4}$$

where g_θ, a parameterized version of g, is determined by the integral of the GNN f_θ in Eq. 1.

To approximate the integration in $\phi_{T,g_\theta}(\boldsymbol{q}^{(t_0)})$ and $\psi_{T,g_\theta,f_\theta}(\boldsymbol{q}_\theta^{(t_0)})$ with parameterized g_θ and f_θ, SEGNO employs an ODE solver to generate a discrete trajectory that approximates the underlying continuous latent trajectory. The time interval T is divided into τ equal sub-intervals, with a timestep $\Delta t = T/\tau$.

Choosing two proper increment functions $\mathcal{G}_1$ and $\mathcal{G}_2$ to approximate the increment of the integral, the relations for the numerical integrators $\Psi_{\Delta t,g,f}$ and $\Phi_{\Delta t,g}$ which approximate $\psi_{\Delta t,g,f}$ and $\phi_{\Delta t,g}$, respectively, are:

$$\begin{aligned}\boldsymbol{q}_\theta^{(t+\Delta t)} = g_\theta(\boldsymbol{q}_\theta^{(t+\Delta t)}, \boldsymbol{h}) = \Psi_{\Delta t,g_\theta,f_\theta}(\boldsymbol{q}_\theta^{(t)}) &= \boldsymbol{q}_\theta^{(t)} + \mathcal{G}_1(f_\theta(\boldsymbol{q}_\theta^{(t)}, \boldsymbol{h}), \Delta t), \\ \boldsymbol{q}_\theta^{(t+\Delta t)} = \Phi_{\Delta t,g_\theta}(\boldsymbol{q}_\theta^{(t)}) &= \boldsymbol{q}_\theta^{(t)} + \mathcal{G}_2(\Psi_{\Delta t,g_\theta,f_\theta}(\boldsymbol{q}_\theta^{(t)}), \Delta t),\end{aligned} \tag{5}$$

For instance, with the increment functions $\mathcal{G}_1(x, y) = \mathcal{G}_2(x, y) = x \times y$, the numerical integrators become the Euler integrators

$$\boldsymbol{q}_\theta^{(t+\Delta t)} = \boldsymbol{q}_\theta^{(t)} + f_\theta(\boldsymbol{q}_\theta^{(t)})\Delta t, \quad \boldsymbol{q}_\theta^{(t+\Delta t)} = \boldsymbol{q}_\theta^{(t)} + \boldsymbol{q}_\theta^{(t+\Delta t)}\Delta t. \tag{6}$$

A comprehensive description of the full algorithm can be found in the original SEGNO paper by Yang Liu et al. [21].

3.3 EGNO

EGNO [22] overcome the limitations of traditional models in capturing temporal correlations along trajectories by directly modeling system dynamics as temporal functions. Predicting dynamics over continuous trajectories reduces the reliance on discrete state transitions. Inspired by FNO [53], EGNO benefits from its desirable properties, including robust discretization convergence guarantees.

The model architecture is made of temporal equivariant convolution layers $\mathcal{T}_\theta$, which are built upon Fourier integration operators $\mathcal{K}_\theta$ to model the temporal correlations. Let $f : D \to \mathcal{G}$ be the input starting function that describes structures $\mathcal{G}^{(t)}$ for time t, i.e., $f(t) = [f_{\mathbf{h}}, f_{\mathbf{Z}}(t)]^{\mathrm{T}}$. Then the convolution layer $\mathcal{T}_\theta$ is implemented as

$$(\mathcal{T}_\theta f)(t) = f(t) + \sigma((\mathcal{K}_\theta f)(t)). \tag{7}$$

The first term in the summation represents the linear component of the original FNO implementation. This part operates without any additional trainable weights, effectively forming a residual connection. This design has proven beneficial in various applications, including operator learning, by facilitating efficient training and preserving essential information.

Unlike traditional EGNNs, which are limited to next-step predictions, this architecture leverages the NO framework to model multiple states from the dynamics simultaneously, thereby capturing temporal correlations

more effectively. Furthermore, EGNO incorporates SE(3)-equivariant temporal convolutions parameterized in the Fourier space, enabling it to learn temporal relationships while maintaining symmetry constraints.

The key innovation lies in implementing $\mathcal{K}_\theta$ using a block diagonal weight matrix $\mathbf{M}_\theta$, defined as

$$(\mathcal{K}_\theta f)(t) = \mathcal{F}^{-1}\left(\begin{bmatrix} \mathbf{M}_\theta^{\mathbf{h}} & \boldsymbol{0} \\ \boldsymbol{0} & \mathbf{M}_\theta^{\mathbf{Z}} \end{bmatrix} \cdot \left(\mathcal{F}\begin{bmatrix} f_{\mathbf{h}} \\ f_{\mathbf{Z}} \end{bmatrix}\right)\right)(t), \tag{8}$$

where $\mathcal{F}$ and $\mathcal{F}^{-1}$ are the direct and inverse fast Fourier transform and $\mathbf{M}_\theta^{\mathbf{h}} \in \mathbb{C}^{I\times k\times k}$ and $\mathbf{M}_\theta^{\mathbf{Z}} \in \mathbb{C}^{I\times m\times m}$ are two complex-valued matrices. Finally, the dimension I for these matrices controls the maximal number of frequency modes and can be tuned.

EGNO employs a stack of equivariant temporal convolution layers combined with equivariant networks. Starting with the current graph state $\mathcal{G}(t)$, the model replicates its features and augments them with time embeddings. The augmented features then pass through L blocks made of a Temporal Convolution Layer, operating on temporal and channel dimensions to capture temporal dependencies, and an EGNN Layer, acting on node and channel dimensions to model spatial interactions within each graph. The final output is a temporal function capable of predicting future dynamics, enabling efficient parallel decoding across a trajectory of states.

Discretizing the time window ΔT into intervals $\{\Delta t_1, \ldots, \Delta t_P\}$, the workflow begins by taking the current state $\mathcal{G}^{(t)}$ as input. The feature map is repeated P times, expanded and passed to the model together with time embeddings of the expected P output snapshots. Both inputs pass through a sequence of blocks composed of temporal convolutions and EGNN layers. In particular, EGNN layers operate solely on the node and channel dimensions of each graph, while treating the temporal dimension as part of the batch dimension. For a detailed description of the complete algorithm, we refer the reader to the original EGNOs paper by Xu et al. [22].

3.4 Experimental setting

The following section will present the experimental setting, outlining the adopted techniques and their primary purposes. We design a comparative evaluation of the overall behavior of both models under different conditions. As such, we do not isolate the impact of individual architectural components (e.g., second-order dynamics, Fourier layers). A detailed ablation study is left for future work.

The primary objective of these experiments is to assess the robustness of both architectures under non-standard conditions. To this end, we first examine their scalability in terms of the number of particles, evaluating both predictive performance and memory efficiency. We then test the models under more unconventional input scenarios, such as multiple or asynchronous inputs, to investigate properties like time invariance. In certain cases, we introduce minor extensions to the original architectural logic to accommodate these settings, while preserving their core design principles. Code for replicating experiments and generating the dataset used is available in the paper repository[1]

3.4.1 Time invariance

As an initial benchmark analysis, we evaluated the time-invariance properties of both models. Specifically, we assessed their ability to make predictions at a different ΔT than the one encountered during training. We will recall this property as *resolution invariance* from now on. For SEGNO, this involves executing the Neural ODE steps a varying number of times. Theoretically, this should have minimal impact on the outcome, as it simply alters the number of integration steps within the model's embedding space. In contrast, EGNO is inherently trained to predict across multiple timesteps simultaneously, as determined by the set of embedded $\{\Delta t_i\}$ intervals.

[1] https://anonymous.4open.science/r/NO-NODE-comparison

In this case, a natural test of time invariance is to provide the model with unseen new values of Δt, despite the original algorithm assuming a fixed time step. While this approach could potentially benefit from the FNO capacity to generalize across time scales, it may also degrade performance when the test time intervals fall significantly outside the training distribution (Fig. 1).

3.4.2 Rollout

When evaluating a model using a rollout technique, several key aspects are assessed to gauge its performance over extended sequences or time horizons. Stability is critical, as models must maintain consistent predictions without errors compounding over time, especially in dynamical systems, where inaccuracies can cascade. The accumulation of errors is another focus, examining whether inaccuracies increase or decrease as the rollout progresses. Long-term consistency is vital, particularly for physical systems, ensuring that predictions respect system constraints such as conserved quantities.

The rollout technique is applied in all experiments, with variations designed to explore the models' capabilities in greater detail. Each experiment uses trajectories consisting of 100 steps, structured so that every macro step includes 10 "inner" steps ($\Delta T = 10$). At each rollout iteration, the first step serves as the input, while the 10th step becomes the output. For EGNO, the entire 10-step trajectory is predicted, with the final step selected as the output. This setup results in a total trajectory length of 100 steps, comprising 10 prediction steps, each spanning a $\Delta T = 10$ time window.

3.4.3 Multiple inputs

In this experiment, the goal was to get an insight into the capabilities of the models to maintain stability in their predictions when receiving more inputs than just the initial one. This variant is applied differently to the two models based on their specific way of handling input timesteps.

SEGNO

In this case, when using multiple inputs, the prediction from last step is aggregated in some way (usually just summed) with the observation of the current step. This process is applied until the specified number of inputs is reached.

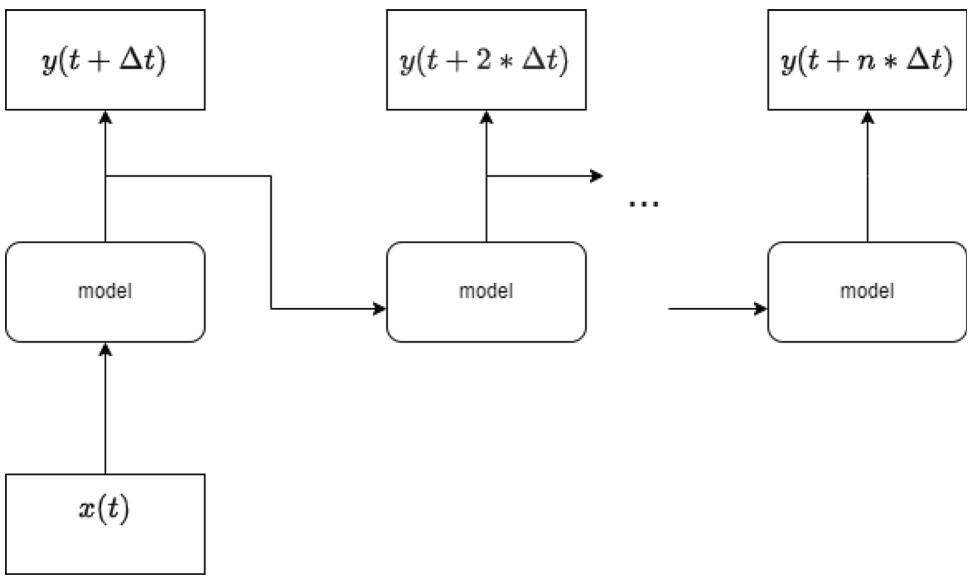


Fig. 1 Schematic representing the rollout technique.

While this approach is standard, our use case represents each input snapshot as a set of positions and velocities. Then, averaging more instances of both separately may lead to a loss of information. Therefore, we designed an updated version in which the aggregation of multiple inputs is achieved using an attention mechanism based on the magnitude of velocity across the input sequence. Instead of summing predictions and ground truth states at each step, we first compute scalar attention weights from the velocity norms and node features, which preserve invariance, at each past time step. These weights are then used to form a convex combination of the predicted observation and the new real one, producing an initial condition that is passed to the model. This passage is performed iteratively for each incoming input state. This attention-based strategy preserves equivariance by ensuring that only invariant inputs contribute to the attention scores, while the final aggregated positions and velocities remain equivariant by construction. Formally, given a sequence of node states $\{G^{(t_0)}, \ldots, G^{(t_n)}\}$, each $G^{(t)}$ being $(\boldsymbol{q}^{(t)}, \boldsymbol{q}^{(t)})$, the model F_θ computes each input after the initial one as a weighted sum:

$$\bar{G}^{(t_n)} = \boldsymbol{\alpha}_{t_n} \cdot \left[G^{(t_n)}, F_\theta\left(G^{(t_{n-1})}\right)\right], \tag{9}$$

where the weights $\boldsymbol{\alpha}_{t_n}$ are derived from a learned function of $\|\boldsymbol{q}^{(t_i)}\|$ and the invariant node features. This approach allows the model to prioritize more informative inputs within the sequence, while still maintaining geometric consistency. As in the previous method, this initialization scheme is used during both training and inference. This strategy is inspired by various algorithms exploring attention-based modules to enhance Neural ODEs and similar models performance [57, 58]. The pseudocode for the strategy that allows multiple inputs for SEGNO is as follows:

Algorithm 1 Training with Velocity Attention Multi-Input Rollout for SEGNO

Algorithm 1 Training with Velocity-Attention Multi-Input Rollout for SEGNO

Require: Ground truth sequence $\{G(t), G(t+\Delta T), \ldots, G(t+(K-1)\Delta T)\}$, model F_θ, attention module $\mathcal{A}_\phi$, number of multi-input steps K, number of rollout iterations N

Extract invariant node features $\{H^{(i)}\}_{i=0}^{K-1}$ and velocities $\{V^{(i)}\}_{i=0}^{K-1}$ from $\{G(t+i\Delta T)\}$

Initialize $X \leftarrow G(t + K\Delta T)$

for $i = K+1$ to N **do**

Predict: $\hat{G}(t + i\Delta T) \leftarrow F_\theta(X)$

Compute attention weights: $\boldsymbol{\alpha}^{(i)} \leftarrow \mathcal{A}_\phi(\{H^{(i)}, V^{(i)}\})$

Compute weighted sum: $X \leftarrow \boldsymbol{\alpha}_{t_n} \cdot \left[\hat{G}\left(t + i\Delta T\right), G\left(t + i\Delta T\right)\right]$

end for

EGNO

For this architecture, the procedure applied to use multiple inputs is different from that of the previous one. In the standard usage of the model, the output time window ΔT is discretized into P points. An input current state $G(t)$ is first repeated its features by P times, then concatenated with the time embeddings, and final fed into the model. When adopting multiple inputs, the time window ΔT, is equally (whenever possible) divided between the input values and then concatenated with time embeddings accordingly. For instance, if two inputs are used, the first $\frac{\Delta T}{2}$ values are equal to the first input and the last $\frac{\Delta T}{2}$ values are equal to the second input. Then, the time embeddings will be computed to reflect this separation. It is important to mention that the original EGNO implementation includes time embeddings solely for the P output timesteps, as all input timesteps are identical. In contrast, we applied a similar strategy to embed the input times, allowing the model to capture this new information effectively. This multiple-input strategy only considers the positions and velocities, while edge

attributes are associated with the last input timestep. This methodology is applied both at training and at validation/test time. The pseudocode is reported in Algorithm 2.

Algorithm 2 Training with Multiple Inputs for EGNO (Trajectory Modeling)

Require: Initial state(s) $\{G_1(t), G_2(t), \ldots, G_K(t)\}$, number of EGNO blocks L, prediction horizon ΔT, number of discretization points P
  Discretize ΔT into P time points: $\{t_1, t_2, \ldots, t_P\}$
  if $K = 1$ **then**
    Repeat $G_1(t)$ for P times: $X \leftarrow \text{repeat}(G_1(t), P)$
  else
    Divide P into K segments of approximately equal length: $P = P_1 + P_2 + \cdots + P_K$
    $X \leftarrow \text{concatenate}\,[\text{repeat}(G_1(t), P_1), \text{repeat}(G_2(t), P_2), \ldots, \text{repeat}(G_K(t), P_K)]$
  end if
  Compute time embeddings $T_e = \text{TimeEmbedding}([t_1, \ldots, t_P])$
  Concatenate input with time embeddings: $X \leftarrow \text{concat}(X, T_e)$
  for $\ell = 1$ to L **do**
    $X \leftarrow \text{EGNOBlock}_\ell(X)$
  end for
  Output predicted trajectory: $\hat{G}(t + \Delta T) \leftarrow X$

3.4.4 Variable time window

When using multiple inputs, the time window Δt (different from ΔT, which is the entire time window of the single prediction step) is kept constant, which means that the time distance between two consecutive inputs is always the same (Note: in the case of SEGNO with multiple inputs $\Delta t = \Delta T$, while in EGNO with multiple inputs Δt is the time distance between steps given as inputs inside the window $\Delta T = 10$). Everything else said in the multiple input section remains true, but in addition to that, in this experiment, the variables Δt are also considered for both models.

Varying the time interval between inputs can reveal how the state of the system changes over different timescales, highlighting the dynamics of slow versus fast-changing components. This can lead to insights into how sensitive the system is to changes over time, potentially identifying which factors or events cause significant shifts.

Moreover, introducing variability in time steps allows the model to handle situations where inputs come at irregular intervals, which is often the case in real-world scenarios (e.g., irregular sampling in sensor data). This can make the model more robust and generalizable as it learns to adapt to a variety of temporal patterns rather than only fixed intervals.

The variable Δt can be incorporated in the two models in a different way. Figure 2 shows a possible approach for EGNO, in which the input P snapshots are not constant repetitions of a single timestamp, but can be any combination of the desired input graphs. The model can understand the temporal meaning of each input snapshot thanks to the use of input time embeddings.

In contrast, SEGNO can manage variable time ranges between inputs by taking different numbers of integration steps to build each prediction of the subsequent snapshot. At the same time, the number of steps will end up being discrete.

SEGNO

As explained before, SEGNO considers just one step as input for each model call and returns just the final prediction. To consider non-uniform sampling, in this experiment, the time window between input and output

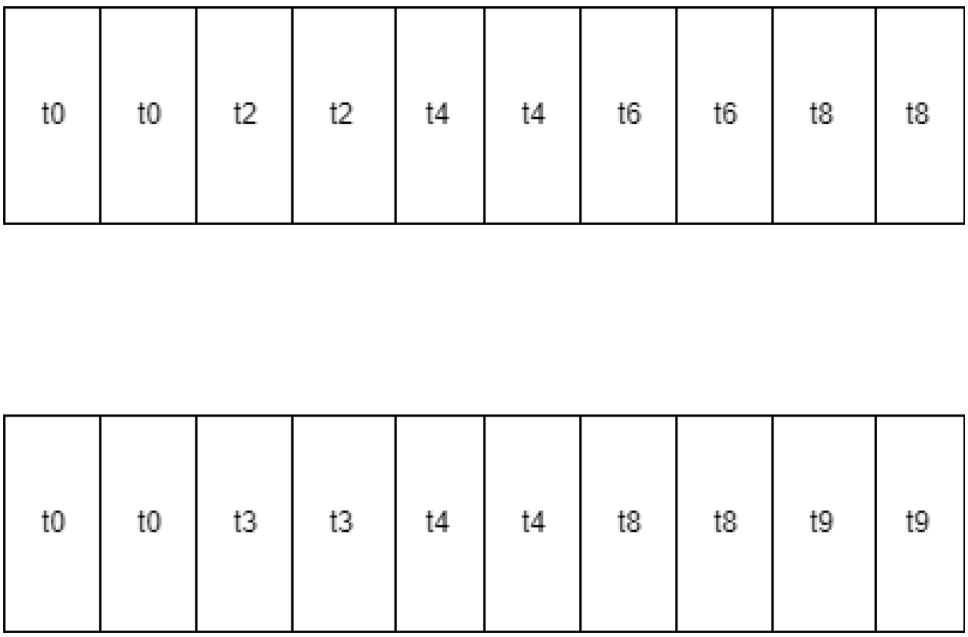


Fig. 2 Schematic representing an example of input given to EGNO when using 5 different inputs (figure on top) and when using multiple inputs with variable Δt (figure on the bottom).

prediction is not fixed but can change along the trajectory (the total length of the trajectory though, stays the same as the number of prediction steps).

EGNO

In the case of EGNO, the model usually takes as input a single step that is then duplicated ΔT times. When using multiple inputs, different steps are given as input to the model. These steps are equispaced, which means that the time distance between two next steps is always the same (we call this Δt) as shown in the upper part of Figure 2. For this experiment, instead, Δt is allowed to change, thus having as an example input what is shown in the lower part of Figure 2). The time embedding of the input is calculated according to the selected timesteps, as was done for the "basic" multiple input experiment.

4 Results

4.1 N-body system simulations

We adopted for the experiments two 3D N-body simulation datasets [45], which comprise multiple trajectories depicting the dynamical system formed by charged particles and interacting through Coulomb forces, and particles with variable mass and gravitational interactions between them. We recall the two datasets as *charged* and *gravity* from now on. The experimental setup considers different time windows (2,5,10), but in the following, the main results reported regard $\Delta T = 10$. Moreover, 3000/2000/2000 trajectories are used for training/validation/testing. For EGNO, uniform discretization is used, with $P = 10$ points in each time window (which means that the model predicts every timestep within the ΔT time window).

We evaluated the results in terms of average Mean Squared Error (MSE) and Energy conservation. This last metric is calculated as the average distance between the expected total energy of the system, which should stay constant along the trajectory. All the metrics are collected with increasing lead time to show the models' capabilities to forecast subsequent system configurations over time from the provided inputs. We repeated all experiments with 10 different initializations and averaged the results to ensure statistical significance. Figure 3 reports a few examples of trajectories jointly with EGNO and SEGNO predictions.

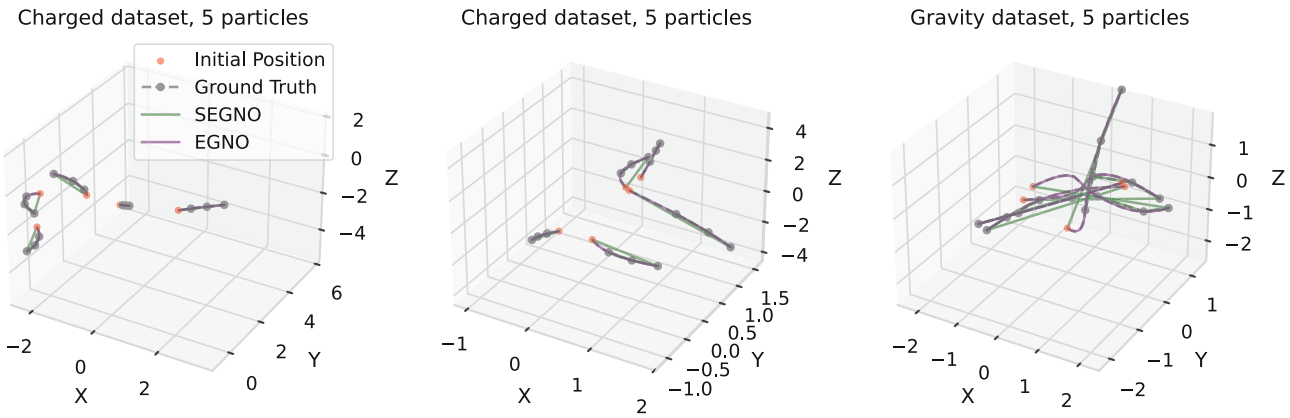


Fig. 3 Sample trajectories with models' predictions for both the gravity and the charged dataset. Ground truth lines are highlighted with a gray point at each ΔT steps.

We conducted all the experiments using one GPU Nvidia A40. We repeated each experiment with an increasing number of particles from 5 to 20 to highlight the scalability of both models. Table 1 shows the required GBs for both models when increasing the number of particles. SEGNO significantly increases GPU usage, rising from approximately 700 MB to around 17 GB. In contrast, EGNO increase is more stable, ranging from 670 MB to 5 GB.

4.2 Time invariance

Figure 4 compares the test loss of SEGNO and EGNO across varying evaluation time window ΔT_{input} and training time window ΔT_{model} configurations. We evaluated the model on the first predicted snapshot without rollout, i.e., using SEGNO's single prediction versus the first of EGNO's P predictions from one forward pass. SEGNO demonstrates a consistent and smooth degradation in performance as the test ΔT_{input} diverges from the training ΔT_{model}, suggesting robustness to timestep shifts. Conversely, EGNO performs well when $\Delta T_{\text{input}} = \Delta T_{\text{model}}$, but exhibits erratic and often catastrophic performance as the test timestep deviates, indicating strong dependence on the trained time resolution. This behavior reflects EGNO's sensitivity to timestep mismatch and highlights SEGNO's more stable generalization across time scales.

4.3 Standard rollout

The baseline training techniques involve using the rollout procedure at test time without modifications to the training procedure of the models' papers. The results for the gravity dataset using 5-particles simulations are shown in Figure 5. The figure presents the aggregated test performance of both models, evaluated using MSE and the correlation between predicted and ground truth values across increasing rollout steps. Each rollout step corresponds to predicting the system state ΔT time steps ahead. Specifically, EGNO produces a prediction for every intermediate timestep between the last input and the target time, whereas SEGNO generates only the final state prediction.

The investigated timesteps involve only the initial part of the trajectory because EGNO quickly diverges beyond this region. SEGNO, on the other hand, remains stable throughout the trajectory for a longer time.

Table 1 GPU memory usage in GB for each model at varying number of particles

# Particles	5	10	20
EGNO	0.67	1.55	5.25
SEGNO	0.70	2.66	17.22

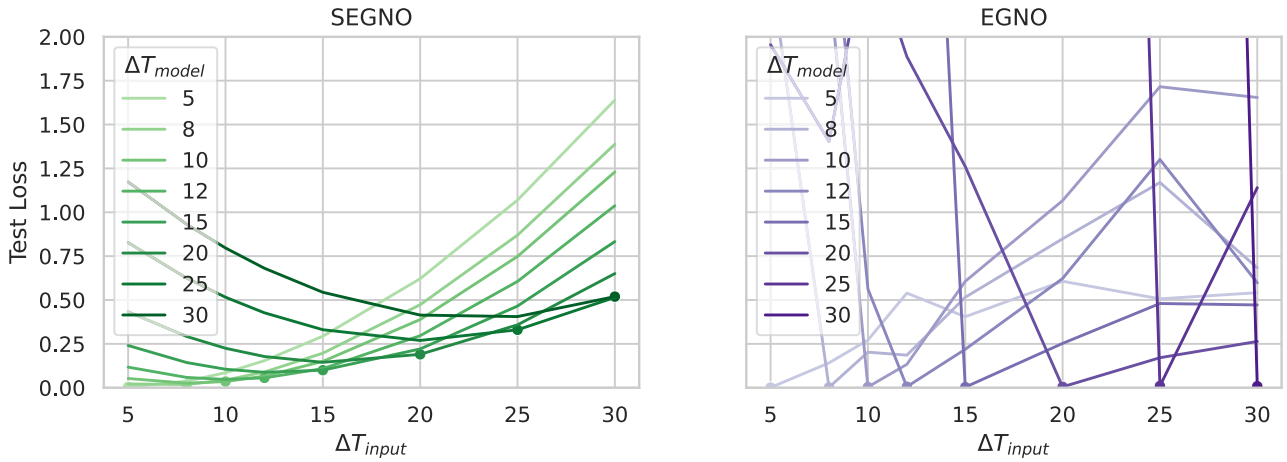


Fig. 4 Test loss as a function of input timestep ΔT_{input} for different model timestep settings ΔT_{model}, comparing SEGNO (left) and EGNO (right). Predictions are evaluated without rollout.

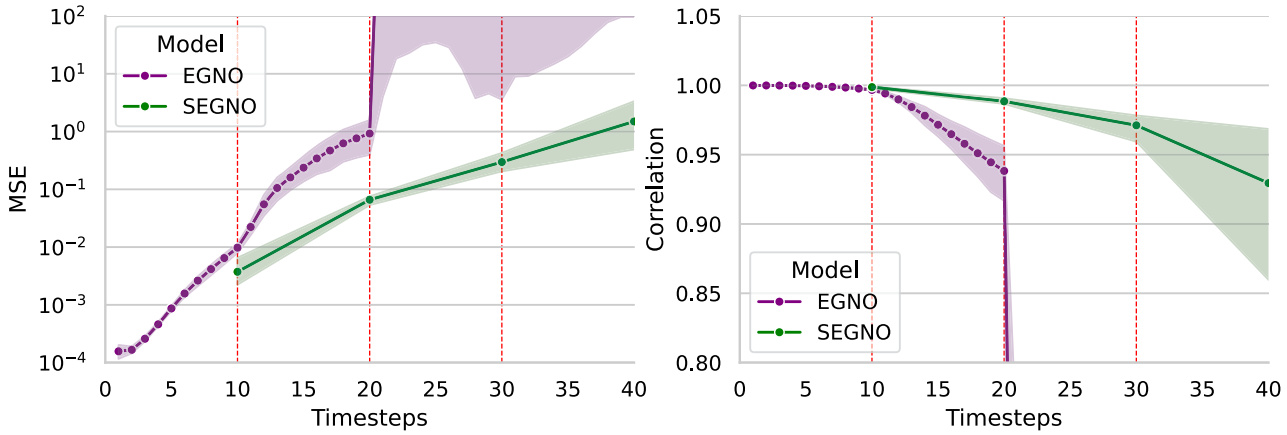


Fig. 5 Trajectories Mean MSE (left in log scale) and Correlation (right) with error calculated along 5 runs, after applying Rollout to standard EGNO and SEGNO on the gravity dataset with 5 particles. The thick points represent the models' calculated points, while the red dashed lines stand for the ΔT intervals, meaning the final prediction after each rollout iteration.

Moreover, after the second rollout iteration, it can be seen that the standard deviation in EGNO starts diverging, indicating that the model is unable to perform further rollout iterations. The rapid degradation of the correlation also reinforces this point.

Regarding MSE, both models start to struggle after the second iteration of the rollout (so after $t = 20$ in this case). However, in contrast to EGNO extreme degradation, SEGNO still maintains some sort of relevance, even though the error starts increasing more and more. This behavior should be expected in this kind of experiment, since, at every further iteration, the errors add up, leading to a misalignment with the ground truth. Collecting all the experiments, Table 2 reports the performance at a variable number of particles for both datasets. EGNO consistently reaches better performance at the beginning of the trajectory, but starts degrading from the second rollout iteration. In contrast, SEGNO tends to provide more stable predictions, especially for a small number of particles, but tends to fail in both datasets when the number of particles increases. SEGNO's inability to capture particle behavior becomes even more evident when examining energy conservation, which exhibits noticeably large errors from the very first rollout step in simulations involving 20 particles.

Table 2 MSE and energy conservation for baseline models. Both metrics are computed at the $1 \cdot \Delta T$ and $2 \cdot \Delta T$. The best results for each number of particles and dataset are highlighted in bold

Dataset	N parts	Model	MSE ΔT	MSE $2\Delta T$	Energy cons. ΔT	Energy cons. $2\Delta T$
charged	5	EGNO	**0.003 (0.000)**	0.235 (0.276)	2.609 (0.582)	9.169 (4.889)
		SEGNO	0.006 (0.000)	**0.045 (0.009)**	**0.794 (0.100)**	**0.971 (0.118)**
	10	EGNO	**0.005 (0.000)**	0.094 (0.030)	5.730 (1.902)	22.904 (11.525)
		SEGNO	0.007 (0.000)	**0.048 (0.016)**	**1.470 (0.384)**	**1.747 (0.484)**
	20	EGNO	**0.006 (0.001)**	**0.155 (0.090)**	**12.565 (5.675)**	**55.867 (36.417)**
		SEGNO	0.021 (0.029)	1.101 (2.333)	1518.853 (3389.566)	1672.595 (3731.609)
gravity	5	EGNO	**0.003 (0.001)**	0.187 (0.144)	5.235 (2.101)	54.144 (93.169)
		SEGNO	0.004 (0.003)	**0.035 (0.008)**	**0.697 (0.310)**	**1.364 (1.328)**
	10	EGNO	**0.007 (0.001)**	**0.565 (0.256)**	**15.827 (5.105)**	**84.635 (75.422)**
		SEGNO	0.063 (0.121)	1.114 (2.003)	122.927 (189.400)	117.314 (156.917)
	20	EGNO	**0.019 (0.001)**	**0.411 (0.107)**	**70.105 (9.027)**	**140.713 (26.334)**
		SEGNO	0.509 (0.956)	9.013 (15.867)	3759.337 (6387.728)	10392.845 (18412.983)

4.4 Multiple inputs

We now evaluate models trained and tested with multiple inputs (MI) to assess whether this approach enhances the stability, improving models' performance. Note that the introduction of multiple inputs varies between the two architectures, leading to different impacts on their results.

Table 3 reports the results of the MI version of EGNO and SEGNO models, comparing standard single-input training to configurations with two and three input states. The table reports the performance in terms of MSE and

Table 3 MSE and energy conservation for the models trained with multiple inputs. Both metric are computed at the first, the fifth and the tenth rollout iterations, respectively. The best results for each number of particles and dataset are highlighted in bold

Dataset	N parts	Model	MSE ΔT	Energy cons. ΔT	MSE improvement	Energy cons. improvement
charged	5	EGNO MI(2)	**2.91e-03 (1.6e-05)**	**3.82 (1.1)**	9.5%	−38.4%
		EGNO MI(3)	3.06e-03 (8.3e-05)	5.12 (0.8)	4.8%	$< -50\%$
		SEGNO MI(2)	0.02 (2.3e-04)	42.90 (15.1)	$< -50\%$	$< -50\%$
		SEGNO MI(3)	0.01 (5.8e-04)	8.01 (0.2)	$< -50\%$	$< -50\%$
	10	EGNO MI(2)	3.88e-03 (6.9e-04)	**7.44 (1.8)**	22.4%	−20.7%
		EGNO MI(3)	**3.75e-03 (6.2e-04)**	8.56 (0.8)	24.9%	−38.9%
		SEGNO MI(2)	0.02 (1.1e-03)	235.50 (155.9)	$< -50\%$	$< -50\%$
		SEGNO MI(3)	0.02 (5.7e-04)	99.64 (51.2)	$< -50\%$	$< -50\%$
gravity	5	EGNO MI(2)	**2.61e-03 (3.3e-04)**	6.39 (2.0)	−7.8%	−21.8%
		EGNO MI(3)	2.72e-03 (1.0e-04)	6.18 (0.9)	−12.2%	−17.9%
		SEGNO MI(2)	0.17 (4.5e-03)	29.14 (8.5)	$< -50\%$	$< -50\%$
		SEGNO MI(3)	0.24 (8.4e-04)	**5.95 (2.9)**	$< -50\%$	$< -50\%$
	10	EGNO MI(2)	**6.65e-03 (2.7e-04)**	21.42 (11.1)	8.4%	−33.8%
		EGNO MI(3)	9.11e-03 (4.4e-04)	25.14 (20.9)	−25.4%	$< -50\%$
		SEGNO MI(2)	0.22 (4.3e-03)	131.68 (46.6)	$< -50\%$	$< -50\%$
		SEGNO MI(3)	0.31 (4.4e-03)	**9.41 (0.6)**	$< -50\%$	$< -50\%$

energy consistency of the first forward pass for each model. Then, the last two columns show the percentage of performance gain concerning the single-input results. For EGNO, the use of multiple inputs leads to improved performance in both MI(2) and MI(3) settings. This improvement is particularly evident in the charged dataset, where MSE consistently decreases. In the gravity dataset, however, the benefit is less pronounced, with MSE values remaining largely comparable to those of the single-input model. Interestingly, while MSE improves with multiple inputs, energy conservation metrics consistently worsen across all settings, indicating a trade-off between predictive accuracy and physical consistency.

In contrast, SEGNO does not benefit from multiple inputs. Both MI(2) and MI(3) settings lead to degraded performance across all metrics and datasets. This deterioration is especially severe in the gravity dataset, where both MSE and energy conservation show substantial declines compared to the single-input baseline. These results suggest that, unlike EGNO, SEGNO is not robust to the inclusion of multiple inputs under the current training strategy, and alternative approaches may be needed to stabilize its predictions.

Finally, EGNO in the MI(2) setting appears to be the best model in terms of overall performance and performance improvements to its baseline.

4.5 Variable time windows

We now consider the model variations when trained with a variable distance between the timesteps. In addition, in this case, the practical adaptation of using variable ΔT differs between the two models, as explained in Section 3.4.4. In practice, we aim to evaluate a task that involves non-uniform sampling to assess the robustness of the two proposed approaches under more challenging conditions. Although this setting may lead to a decline in overall performance, our focus is on testing model stability. To accommodate this variant of the original task, the training procedure has been adjusted accordingly.

Moreover, in EGNO, the concept of variable ΔT is related to the distance between the different inputs provided, more than the actual number of timesteps produced. Given the theoretical foundation of neural operators, which exhibits discretization invariance to some extent, it is interesting to analyze this aspect. Especially, it is worth noting that such a property starts emerging when the model's training already includes multiple time windows.

Results of the variable ΔT (VDT) setting are summarized in Table 4. Starting with EGNO, the results on the charged dataset remain comparable to the baseline for both $n = 5$ and $n = 10$ in terms of MSE, suggesting a degree of robustness to this variation. However, energy conservation deteriorates, indicating a reduced ability to maintain physical consistency in this setting. In contrast, performance on the gravity dataset worsens, with both MSE and energy consistency affected. This suggests that EGNO may struggle to generalize under variable ΔT conditions in more complex scenarios.

SEGNO, while still underperforming relative to EGNO, shows a somewhat unexpected behavior. Although the overall performance remains poor, results under VDT are consistently better than those observed with the fixed-window MI training. This improvement, albeit modest, points to a potential benefit in generalization when training with a variable input window. It suggests that introducing temporal variability during training might help SEGNO develop a more flexible internal representation, even if the model architecture itself is not particularly well-suited to the task.

5 Discussion and conclusion

This work compared EGNO and SEGNO, two recent deep learning approaches to modeling n-body systems, two recent deep learning architectures designed for modeling n-body dynamical systems. We conducted a comprehensive experimental analysis to evaluate their performance under different conditions, with a particular focus on long-term prediction stability and precision. To provide a concise overview of the practical differences between

Table 4 MSE and energy conservation for the models trained with multiple inputs and variable ΔT. Both metric are computed at the first, the fifth and the tenth rollout iterations, respectively. The best results for each number of particles and dataset are highlighted in bold

Dataset	N parts	Model	MSE ΔT	Energy cons. ΔT	MSE improvement	Energy cons. improvement
charged	5	EGNO MI(2) VDT	**2.91e-03 (1.6e-05)**	**3.82 (1.1)**	−35.6%	−29.4%
		EGNO MI(3) VDT	3.06e-03 (8.3e-05)	5.12 (0.8)	−22.6%	$< -50\%$
		SEGNO MI(2) VDT	0.02 (2.3e-04)	42.90 (15.1)	$< -50\%$	$< -50\%$
		SEGNO MI(3) VDT	0.01 (5.8e-04)	8.01 (0.2)	$< -50\%$	$< -50\%$
	10	EGNO MI(2) VDT	3.88e-03 (6.9e-04)	**7.44 (1.8)**	−5.0%	$< -50\%$
		EGNO MI(3) VDT	**3.75e-03 (6.2e-04)**	8.56 (0.8)	1.6%	$< -50\%$
		SEGNO MI(2) VDT	0.02 (1.1e-03)	235.50 (155.9)	$< -50\%$	$< -50\%$
		SEGNO MI(3) VDT	0.02 (5.7e-04)	99.64 (51.2)	$< -50\%$	$< -50\%$
gravity	5	EGNO MI(2) VDT	**2.61e-03 (3.3e-04)**	6.39 (2.0)	$< -50\%$	−48.7%
		EGNO MI(3) VDT	2.72e-03 (1.0e-04)	6.18 (0.9)	$< -50\%$	−35.9%
		SEGNO MI(2) VDT	0.17 (4.5e-03)	29.14 (8.5)	$< -50\%$	$< -50\%$
		SEGNO MI(3) VDT	0.24 (8.4e-04)	**5.95 (2.9)**	$< -50\%$	$< -50\%$
	10	EGNO MI(2) VDT	**6.65e-03 (2.7e-04)**	21.42 (11.1)	$< -50\%$	7.8%
		EGNO MI(3) VDT	9.11e-03 (4.4e-04)	25.14 (20.9)	$< -50\%$	$< -50\%$
		SEGNO MI(2) VDT	0.22 (4.3e-03)	131.68 (46.6)	$< -50\%$	$< -50\%$
		SEGNO MI(3) VDT	0.31 (4.4e-03)	**9.41 (0.6)**	$< -50\%$	26.7%

the models, Table 5 summarizes their comparative performance and design trade-offs across key dimensions such as scalability, stability, and data efficiency.

The experiments involved applying the rollout technique to the models, along with specific variations designed to stress-test their performance. These variations included providing multiple inputs to examine whether this approach could enhance stability in extended steps, as well as implementing variable ΔT values.

For SEGNO, the variable setup of ΔT posed a uniform sampling prediction task, whereas for EGNO, it involved using input time steps with varying distances within the initial prediction step. These modifications required adjustments to both the training and testing procedures to ensure consistency and to effectively evaluate the models' adaptability to these changes.

Table 5 High-level comparison of SEGNO and EGNO based on experimental results

Aspect	SEGNO (neural ODE)	EGNO (neural operator)
Parameter count	Generally lower as the Neural ODE paradigm reuses modules via integration	Higher; explicit multi-step modeling with spectral layers
Extrapolation ability	Stable over long rollouts; robust to error accumulation	Accurate only in early steps; rapidly degrades afterward
Data efficiency	Lower; sensitive to data quantity and input variation	Higher; benefits from richer and irregular input sequences, resulting in more stable outcomes with simulations involving a higher number of particles
Resolution invariance	Limited generalization to unseen Δt	It inherits discretization invariance from the operator design, but these benefits manifest only with specifically designed training strategies
Physical consistency	Better for small systems; breaks with more particles	More stable but less accurate in conservation laws
Best use case	Long-term forecasting with known physics	Short-term precision and diverse training regimes

Our experiments revealed that EGNO struggles to maintain stable predictions after a few rollout iterations, whereas SEGNO demonstrated superior stability over longer trajectories. However, in the early stages of the trajectory, where EGNO remains stable, it outperforms SEGNO in terms of average MSE and energy conservation. Moreover, SEGNO struggles with simulations involving more particles, obtaining poor results in terms of both metrics from the first forward pass.

Introducing multiple inputs during training did not provide extreme performance improvements for either model based on the metrics evaluated. Nevertheless, the proposed strategy enhanced EGNO performance across most metrics. In contrast, the results for SEGNO consistently decreased.

Finally, testing with varying prediction step sizes confirmed EGNO's superior performance during the initial part of the trajectory. The observed robustness to changes in Δt suggests that EGNO benefits from architectural features inspired by Neural Operators that promote a certain degree of resolution-independence. However, we found that this property is highly dependent on the model's ability to generalize to unseen time intervals. Further investigation may provide deeper insights into the model's behavior across varying temporal resolutions.

Overall, the standard SEGNO showed to be a more robust architecture to model long multi-step trajectories, but it seems that the training strategies adopted were not beneficial. EGNO instead showed less robustness in maintaining stability after a few rollout iterations, but in the portion of the trajectory along which it was reliable, it showed to be more precise than SEGNO on most metrics, also because it was able to benefit from the training techniques adopted.

This work lays the foundation for future exploration in neural modeling of complex dynamical systems. Future work could investigate the scalability of SEGNO and EGNO on higher-particle-count systems. Preliminary trials indicate that EGNO becomes significantly more computationally demanding as the particle count increases, highlighting the challenges of scalability. One promising direction to enhance the model's robustness and potentially reduce computational costs is the integration of physics-informed techniques, such as those employed in VINO [59]. These methods could improve generalization and physical consistency, especially in scenarios where the governing equations are partially known or can be estimated.

Additionally, testing both models on irregularly sampled, multi-modal datasets could offer insights into their generalization capabilities under real-world conditions. Another promising direction involves developing hybrid models that combine SEGNO's long-term stability with EGNO's short-term precision, potentially leveraging their complementary strengths.

Finally, enhancing SEGNO's performance by revisiting its training strategies—for instance, through more effective aggregation of multiple inputs or alternative architectural modifications—could unlock further gains in accuracy without sacrificing robustness.

Funding Open access funding provided by Politecnico di Torino within the CRUI-CARE Agreement.

Data availability Directly available.

Code availability Available.

Declarations

Conflict of interest None.

Ethical approval Not applicable.

Consent to participate Not applicable.

Consent for publication Not applicable.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/.

References

1. Frenkel D, Smit B (2023) Understanding Molecular Simulation: from Algorithms to Applications. Elsevier, Amsterdam
2. Springel V (2005) The cosmological simulation code gadget-2. Mon Not R Astron Soc 364(4):1105–1134
3. Liu M (2010) Liu G Smoothed particle hydrodynamics (sph): an overview and recent developments. Archi Comput Methods Eng 17:25–76
4. Allen MP, Tildesley DJ (2017) Computer Simulation of Liquids. Oxford University Press, Oxford
5. Schütt KT, Arbabzadah F, Chmiela S, Müller KR (2017) Tkatchenko A Quantum-chemical insights from deep tensor neural networks. Nat Commun 8(1):13890
6. Cranmer M, Sanchez Gonzalez A, Battaglia P, Xu R, Cranmer K, Spergel D (2020) Ho S Discovering symbolic models from deep learning with inductive biases. Adv Neural Inf Process Syst 33:17429–17442
7. Sanchez-Gonzalez A, Godwin J, Pfaff T, Ying R, Leskovec J, Battaglia P (2020) Learning to simulate complex physics with graph networks. In: International Conference on Machine Learning, pp. 8459–8468. PMLR
8. Greydanus S, Dzamba M, Yosinski J (2019) Hamiltonian neural networks. Advances in neural information processing systems **32**
9. Battaglia PW, Hamrick JB, Bapst V, Sanchez-Gonzalez A, Zambaldi V, Malinowski M, Tacchetti A, Raposo D, Santoro A, Faulkner R, Gulcehre C, Song F, Ballard A, Gilmer J, Dahl G, Vaswani A, Allen K, Nash C, Langston V, Dyer C, Heess N, Wierstra D, Kohli P, Botvinick M, Vinyals O, Li Y, Pascanu R (2018)Relational inductive biases, deep learning, and graph networks. arXiv:1806.01261 [cs, stat]. 2651 citations (Semantic Scholar/arXiv) [2024-04-18] arXiv:1806.01261. Accessed 2021-12-16
10. Venkatachalapathy P (2023) Mallikarjunaiah S A feedforward neural network framework for approximating the solutions to nonlinear ordinary differential equations. Neural Comput Appl 35(2):1661–1673
11. Takamoto M, Praditia T, Leiteritz R, MacKinlay D, Alesiani F, Pflüger D (2022) Niepert M Pdebench: an extensive benchmark for scientific machine learning. Adv Neural Inf Process Syst 35:1596–1611
12. Monaco S (2023) Apiletti D Training physics-informed neural networks: One learning to rule them all? Results in Eng 18:101023
13. Chmiela S, Tkatchenko A, Sauceda HE, Poltavsky I, Schütt KR T, Müller K (2017) Machine learning of accurate energy-conserving molecular force fields. Sci Adv 3(5):1603015
14. Siebenmorgen T, Menezes F, Benassou S, Merdivan E, Didi K, Mourão ASD, Kitel R, Liò P, Kesselheim S, Piraud M, Theis FJ, Sattler M, Popowicz GM Misato (2024) machine learning dataset of protein–ligand complexes for structure-based drug discovery. Nat Comput Sci 1–12
15. Turk MJ, Smith BD, Oishi JS, Skory S, Skillman SW, Abel T, Norman ML (2010) A multi-code analysis toolkit for astrophysical simulation data. Astrophys J Suppl Ser 192(1):9
16. Chapon D, Hennebelle P (2024) The galactica database: an open, generic and versatile tool for the dissemination of simulation data in astrophysics. arXiv preprint arXiv:2411.08647
17. Von Rueden L, Mayer S, Beckh K, Georgiev B, Giesselbach S, Heese R, Kirsch B, Pfrommer J, Pick A, Ramamurthy R (2021) Informed machine learning-a taxonomy and survey of integrating prior knowledge into learning systems. IEEE Trans Knowl Data Eng 35(1):614–633
18. Monaco S, Apiletti D (2022) Malnati G Theory-guided deep learning algorithms: an experimental evaluation. Electronics 11(18):2850
19. Chen TQ, Rubanova Y, Bettencourt J, Duvenaud D (2018) Neural ordinary differential equations. CoRR **abs/1806.07366** arXiv:1806.07366
20. Kovachki N, Li Z, Liu B, Azizzadenesheli K, Bhattacharya K, Stuart A (2023) Anandkumar A Neural operator: Learning maps between function spaces with applications to pdes. J Mach Learn Res 24(89):1–97
21. Liu Y, Cheng J, Zhao H, Xu T, Zhao P, Tsung F, Li J, Rong Y (2024) SEGNO: Generalizing Equivariant Graph Neural Networks with Physical Inductive Biases. arXiv:https://arxiv.org/abs/2308.13212
22. Xu M, Han J, Lou A, Kossaifi J, Ramanathan A, Azizzadenesheli K, Leskovec J, Ermon S, Anandkumar A (2024) Equivariant Graph Neural Operator for Modeling 3D Dynamics. arXiv:https://arxiv.org/abs/2401.11037
23. Wu Z, Pan S, Chen F, Long G, Zhang C, Yu PSA (2019) comprehensive survey on graph neural networks. CoRR **abs/1901.00596** arXiv:1901.00596

24. Bronstein MM, Bruna J, Cohen T, Veličković P (2021) Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. arXiv preprint arXiv:2104.13478
25. Rippel O, Snoek J, Adams RP (2015) Spectral representations for convolutional neural networks. Advances in neural information processing systems **28**
26. Veličković P, Cucurull G, Casanova A, Romero A, Liò P, Bengio Y (2018) Graph attention networks. In: International Conference on Learning Representations
27. Rusch TK, Bronstein MM, Mishra SA (2023) survey on oversmoothing in graph neural networks. arXiv preprint arXiv:2303.10993
28. Yu B, Yin H, Zhu Z (2017) Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting. arXiv preprint arXiv:1709.04875
29. Skarding J, Gabrys B (2021) Musial K Foundations and modeling of dynamic networks using dynamic graph neural networks: A survey. IEEE Access 9:79143–79168. https://doi.org/10.1109/access.2021.3082932
30. Zhu Y, Lyu F, Hu C, Chen X, Liu X (2022) Encoder-Decoder Architecture for Supervised Dynamic Graph Learning: A Survey. arXiv:https://arxiv.org/abs/2203.10480
31. Yang M, Zhou M, Kalander M, Huang Z, King I (2021) Discrete-time temporal network embedding via implicit hierarchical learning in hyperbolic space. In: Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining. KDD '21, pp. 1975–1985. ACM, ???. https://doi.org/10.1145/3447548.3467422
32. Sun L, Zhang Z, Zhang J, Wang F, Peng H, Su S, Yu PS (2021) Hyperbolic Variational Graph Neural Network for Modeling Dynamic Graphs. arXiv:https://arxiv.org/abs/2104.02228
33. Hajiramezanali E, Hasanzadeh A, Duffield N, Narayanan KR, Zhou M, Qian X (2020) Variational Graph Recurrent Neural Networks. arXiv:https://arxiv.org/abs/1908.09710
34. Sankar A, Wu Y, Gou L, Zhang W, Yang H (2019) Dynamic Graph Representation Learning via Self-Attention Networks. arXiv:https://arxiv.org/abs/1812.09430
35. Chang X, Liu X, Wen J, Li S, Fang Y, Song L, Qi Y (2020) Continuous-time dynamic graph learning via neural interaction processes. In: Proceedings of the 29th ACM International Conference on Information & Knowledge Management
36. Huang Z, Sun Y, Wang W (2021) Coupled graph ode for learning interacting system dynamics. In: Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining
37. Li Z, Kovachki N, Azizzadenesheli K, Liu B, Bhattacharya K, Stuart A, Anandkumar A (2020) Neural operator: Graph kernel network for partial differential equations. arXiv preprint arXiv:2003.03485
38. Thomas N, Smidt T, Kearnes S, Yang L, Li L, Kohlhoff K, Riley P (2018) Tensor field networks: Rotation- and translation-equivariant neural networks for 3D point clouds. arXiv:https://arxiv.org/abs/1802.08219
39. Fuchs FB, Worrall DE, Fischer V, Welling M (2020) SE(3)-Transformers: 3D Roto-Translation Equivariant Attention Networks. arXiv:https://arxiv.org/abs/2006.10503
40. Finzi M, Stanton S, Izmailov P, Wilson AG (2020) Generalizing convolutional neural networks for equivariance to lie groups on arbitrary continuous data. In: III H.D, Singh A. (eds.) Proceedings of the 37th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 119, pp. 3165–3176. PMLR, ???. https://proceedings.mlr.press/v119/finzi20a.html
41. Horie M, Morita N, Hishinuma T, Ihara Y, Mitsume N (2021) Isometric Transformation Invariant and Equivariant Graph Convolutional Networks. arXiv:https://arxiv.org/abs/2005.06316
42. Köhler J, Klein L, Noé F (2019) Equivariant Flows: sampling configurations for multi-body systems with symmetric energies. arXiv:https://arxiv.org/abs/1910.00753
43. Köhler J, Klein L, Noé F (2020) Equivariant Flows: Exact Likelihood Generative Learning for Symmetric Densities. arXiv:https://arxiv.org/abs/2006.02425
44. Keriven N, Peyré G (2019) Universal invariant and equivariant graph neural networks. Advances in Neural Information Processing Systems **32**
45. Satorras VG, Hoogeboom E, Welling M (2022) E(n) Equivariant Graph Neural Networks. arXiv:https://arxiv.org/abs/2102.09844
46. Monaco S, Barresi S, Apiletti D (2023) Lorentz-invariant augmentation for high-energy physics. Communications in Computer and Information Science. Springer
47. Battaglia P.W, Pascanu R, Lai M, Rezende D.J, Kavukcuoglu K (2016) Interaction networks for learning about objects, relations and physics. CoRR **abs/1612.00222**arXiv:1612.00222
48. Gruver N, Finzi M, Stanton S, Wilson AG (2022) Deconstructing the inductive biases of hamiltonian neural networks. arXiv preprint arXiv:2202.04836
49. Norcliffe A, Bodnar C, Day B, Simidjievski N (2020) Liò P On second order behaviour in augmented neural odes. Adv Neural Inf Process Syst 33:5911–5921
50. Poli M, Massaroli S, Park J, Yamashita A, Asama H, Park J (2019) Graph neural ordinary differential equations. arXiv preprint arXiv:1911.07532
51. Sanchez-Gonzalez A, Bapst V, Cranmer K, Battaglia P (2019) Hamiltonian graph networks with ode integrators. arXiv preprint arXiv:1909.12790

52. Bishnoi S, Bhattoo R, Ranu S, Krishnan N (2022) Enhancing the inductive biases of graph neural ode for modeling dynamical systems. arXiv preprint arXiv:2209.10740
53. Li Z, Kovachki N, Azizzadenesheli K, Liu B, Bhattacharya K, Stuart A, Anandkumar A (2021) Fourier Neural Operator for Parametric Partial Differential Equations. arXiv:https://arxiv.org/abs/2010.08895
54. Yang Y, Gao AF, Castellanos JC, Ross ZE, Azizzadenesheli K, Clayton RW (2021) Seismic wave propagation and inversion with Neural Operators. arXiv:https://arxiv.org/abs/2108.05421
55. Wen G, Li Z.-Y, Long Q, Azizzadenesheli K, Anandkumar A, Benson SM (2022) Accelerating carbon capture and storage modeling using fourier neural operators. ArXiv **abs/2210.17051**
56. Zheng H, Nie W, Vahdat A, Azizzadenesheli K, Anandkumar A (2023) Fast Sampling of Diffusion Models via Operator Learning. arXiv:https://arxiv.org/abs/2211.13449
57. Li J, Zhu Z (2023) Neural lad: a neural latent dynamics framework for times series modeling. Adv Neural Inf Process Syst 36:17345–17356
58. Gao P, Yang X, Zhang R, Huang K, Goulermas JY (2022) Explainable tensorized neural ordinary differential equations for arbitrary-step time series prediction. IEEE Trans Knowl Data Eng 35(6):5837–5850
59. Eshaghi MS, Anitescu C, Thombre M, Wang Y, Zhuang X, Rabczuk T (2025) Variational physics-informed neural operator (vino) for solving partial differential equations. Comput Methods Appl Mech Eng 437:117785

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

Authors and Affiliations

Matteo Celia[1] · Simone Monaco[1] · Daniele Apiletti[1]

✉ Simone Monaco
simone.monaco@polito.it

[1] Department of Control and Computer Engineering, Politecnico di Torino, Corso Castelfidardo, 39, Torino, Italy