References

[1] Walter A Strauss. *Partial differential equations: An introduction*. John Wiley & Sons, 2007.

[2] Lu Lu, Pengzhan Jin, and George Em Karniadakis. Deeponet: Learning nonlinear operators for identifying differential equations based on the universal approximation theorem of operators. *arXiv preprint arXiv:1910.03193*, 2019.

[3] 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(3):218–229, 2021.

[4] Lu Lu, Xuhui Meng, Shengze Cai, Zhiping Mao, Somdatta Goswami, Zhongqiang Zhang, and George Em Karniadakis. A comprehensive and fair comparison of two neural operators (with practical extensions) based on fair data. *Computer Methods in Applied Mechanics and Engineering*, 393:114778, 2022.

[5] 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.

[6] Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Neural operator: Graph kernel network for partial differential equations. *arXiv preprint arXiv:2003.03485*, 2020.

[7] Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Andrew Stuart, Kaushik Bhattacharya, and Anima Anandkumar. Multipole graph neural operator for parametric partial differential equations. *Advances in Neural Information Processing Systems*, 33:6755–6766, 2020.

[8] 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.

[9] Filipe De Avila Belbute-Peres, Thomas Economon, and Zico Kolter. Combining differentiable pde solvers and graph neural networks for fluid flow prediction. In *international conference on machine learning*, pages 2402–2411. PMLR, 2020.

[10] Marten Lienen and Stephan Günnemann. Learning the dynamics of physical systems from sparse observations with finite element networks. In *International Conference on Learning Representations*, 2022.

[11] 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.

[12] Susanne C Brenner, L Ridgway Scott, and L Ridgway Scott. *The mathematical theory of finite element methods*, volume 3. Springer, 2008.

[13] Weizhang Huang and Robert D Russell. *Adaptive moving mesh methods*, volume 174. Springer Science & Business Media, 2010.

[14] Johannes Brandstetter, Daniel E. Worrall, and Max Welling. Message passing neural PDE solvers. In *International Conference on Learning Representations*, 2022.

[15] Gaurav Gupta, Xiongye Xiao, and Paul Bogdan. Multiwavelet-based operator learning for differential equations. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021.

[16] Shuhao Cao. Choose a transformer: Fourier or galerkin. *Advances in Neural Information Processing Systems*, 34, 2021.

[17] Andrew Jaegle, Felix Gimeno, Andy Brock, Oriol Vinyals, Andrew Zisserman, and Joao Carreira. Perceiver: General perception with iterative attention. In *International Conference on Machine Learning*, pages 4651–4664. PMLR, 2021.

[18] Yao-Hung Hubert Tsai, Shaojie Bai, Makoto Yamada, Louis-Philippe Morency, and Ruslan Salakhutdinov. Transformer dissection: A unified understanding of transformer's attention via the lens of kernel. *arXiv preprint arXiv:1908.11775*, 2019.

[19] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[20] Xinliang Liu, Bo Xu, and Lei Zhang. Ht-net: Hierarchical transformer based operator learning model for multiscale pdes. *arXiv preprint arXiv:2210.10890*, 2022.

[21] Ruchi Guo, Shuhao Cao, and Long Chen. Transformer meets boundary value inverse problems. In *The Eleventh International Conference on Learning Representations*, 2023.

[22] Joaquin Quiñonero-Candela and Carl Edward Rasmussen. A unifying view of sparse approximate gaussian process regression. *Journal of Machine Learning Research*, 6(65):1939–1959, 2005.

[23] Edward Snelson and Zoubin Ghahramani. Sparse gaussian processes using pseudo-inputs. In Y. Weiss, B. Schölkopf, and J. Platt, editors, *Advances in Neural Information Processing Systems*, volume 18. MIT Press, 2005.

[24] Michalis Titsias. Variational learning of inducing variables in sparse gaussian processes. In David van Dyk and Max Welling, editors, *Proceedings of the Twelth International Conference on Artificial Intelligence and Statistics*, volume 5 of *Proceedings of Machine Learning Research*, pages 567–574, Hilton Clearwater Beach Resort, Clearwater Beach, Florida USA, 16–18 Apr 2009. PMLR.

[25] Juho Lee, Yoonho Lee, Jungtaek Kim, Adam Kosiorek, Seungjin Choi, and Yee Whye Teh. Set transformer: A framework for attention-based permutation-invariant neural networks. In *International Conference on Machine Learning*, pages 3744–3753. PMLR, 2019.

[26] Yujin Tang and David Ha. The sensory neuron as a transformer: Permutation-invariant neural networks for reinforcement learning. *Advances in Neural Information Processing Systems*, 34, 2021.

[27] Andrew Jaegle, Sebastian Borgeaud, Jean-Baptiste Alayrac, Carl Doersch, Catalin Ionescu, David Ding, Skanda Koppula, Daniel Zoran, Andrew Brock, Evan Shelhamer, Olivier J Henaff, Matthew Botvinick, Andrew Zisserman, Oriol Vinyals, and Joao Carreira. Perceiver IO: A general architecture for structured inputs & outputs. In *International Conference on Learning Representations*, 2022.

[28] Richa Rastogi, Yair Schiff, Alon Hacohen, Zhaozhi Li, Ian Lee, Yuntian Deng, Mert R. Sabuncu, and Volodymyr Kuleshov. Semi-parametric inducing point networks and neural processes. In *The Eleventh International Conference on Learning Representations*, 2023.

[29] Alvaro Sanchez-Gonzalez, Jonathan Godwin, Tobias Pfaff, Rex Ying, Jure Leskovec, and Peter Battaglia. Learning to simulate complex physics with graph networks. In *International conference on machine learning*, pages 8459–8468. PMLR, 2020.

[30] Tobias Pfaff, Meire Fortunato, Alvaro Sanchez-Gonzalez, and Peter Battaglia. Learning mesh-based simulation with graph networks. In *International Conference on Learning Representations*, 2021.

[31] John Guibas, Morteza Mardani, Zongyi Li, Andrew Tao, Anima Anandkumar, and Bryan Catanzaro. Adaptive fourier neural operators: Efficient token mixers for transformers. *arXiv preprint arXiv:2111.13587*, 2021.

[32] Jaideep Pathak, Shashank Subramanian, Peter Harrington, Sanjeev Raja, Ashesh Chattopadhyay, Morteza Mardani, Thorsten Kurth, David Hall, Zongyi Li, Kamyar Azizzadenesheli, et al. Fourcastnet: A global data-driven high-resolution weather model using adaptive fourier neural operators. *arXiv preprint arXiv:2202.11214*, 2022.

[33] Yunyang Xiong, Zhanpeng Zeng, Rudrasis Chakraborty, Mingxing Tan, Glenn Fung, Yin Li, and Vikas Singh. Nyströmformer: A nystöm-based algorithm for approximating self-attention. In *Proceedings of the... AAAI Conference on Artificial Intelligence. AAAI Conference on Artificial Intelligence*, volume 35, page 14138. NIH Public Access, 2021.

[34] Krzysztof Marcin Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Quincy Davis, Afroz Mohiuddin, Lukasz Kaiser, David Benjamin Belanger, Lucy J Colwell, and Adrian Weller. Rethinking attention with performers. In *International Conference on Learning Representations*, 2021.

[35] 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 *European conference on computer vision*, pages 405–421. Springer, 2020.

[36] Matthew Tancik, Pratul Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. *Advances in Neural Information Processing Systems*, 33, 2020.

[37] Vincent Sitzmann, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. *Advances in Neural Information Processing Systems*, 33, 2020.

[38] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[39] 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.

[40] Zijie Li, Kazem Meidani, and Amir Barati Farimani. Transformer for partial differential equations' operator learning. *Transactions on Machine Learning Research*, 2023.

[41] Yuan Yin, Matthieu Kirchmeyer, Jean-Yves Franceschi, Alain Rakotomamonjy, and Patrick Gallinari. Continuous pde dynamics forecasting with implicit neural representations. *arXiv preprint arXiv:2209.14855*, 2022.

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

[43] Maziar Raissi, Paris Perdikaris, and George E 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.

[44] George Em Karniadakis, Ioannis G Kevrekidis, Lu Lu, Paris Perdikaris, Sifan Wang, and Liu Yang. Physics-informed machine learning. *Nature Reviews Physics*, 3(6):422–440, 2021.

[45] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017.

[46] Ali Kashefi, Davis Rempe, and Leonidas J. Guibas. A point-cloud deep learning framework for prediction of fluid flow fields on irregular geometries. *Physics of Fluids*, 33(2):027104, 2021.

[47] Ali Kashefi and Tapan Mukerji. Physics-informed pointnet: A deep learning solver for steady-state incompressible flows and thermal fields on multiple sets of irregular geometries. *J. Comput. Phys.*, 468, nov 2022.

[48] Chiyu "Max" Jiang, Soheil Esmaeilzadeh, Kamyar Azizzadenesheli, Karthik Kashinath, Mustafa Mustafa, Hamdi A. Tchelepi, Philip Marcus, Prabhat, and Anima Anandkumar. Meshfreeflownet: A physics-constrained deep continuous space-time super-resolution framework. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 2020.

[49] Mathis Bode, Michael Gauding, Zeyu Lian, Dominik Denker, Marco Davidovic, Konstantin Kleinheinz, Jenia Jitsev, and Heinz Pitsch. Using physics-informed enhanced super-resolution generative adversarial networks for subfilter modeling in turbulent reactive flows. *Proceedings of the Combustion Institute*, 38(2):2617–2625, 2021.

[50] Shaowu Pan, Steven L Brunton, and J Nathan Kutz. Neural implicit flow: a mesh-agnostic dimensionality reduction paradigm of spatio-temporal data. *arXiv preprint arXiv:2204.03216*, 2022.

[51] David Ha, Andrew Dai, and Quoc V Le. Hypernetworks. *arXiv preprint arXiv:1609.09106*, 2016.

[52] Tianping Chen and Hong Chen. Universal approximation to nonlinear operators by neural networks with arbitrary activation functions and its application to dynamical systems. *IEEE Transactions on Neural Networks*, 6(4):911–917, 1995.

[53] Tapas Tripura and Souvik Chakraborty. Wavelet neural operator for solving parametric partial differential equations in computational mechanics problems. *Computer Methods in Applied Mechanics and Engineering*, 404:115783, 2023.

[54] Yanshuai Cao, Marcus A Brubaker, David J Fleet, and Aaron Hertzmann. Efficient optimization for sparse gaussian process regression. *Advances in Neural Information Processing Systems*, 26, 2013.

[55] Timothy Nguyen, Zhourong Chen, and Jaehoon Lee. Dataset meta-learning from kernel ridge-regression. In *International Conference on Learning Representations*, 2021.

[56] Juncai He and Jinchao Xu. Mgnet: A unified framework of multigrid and convolutional neural network. *Science china mathematics*, 62:1331–1354, 2019.

[57] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.

[58] 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.

[59] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019.

A Derivation of equation 4

Here is a brief explanation of the approximation with the integral for the cross-attention mechanism, where the softmax for the attention matrix is ignored for simplicity.

$$
\begin{aligned}
\mathit{Attn}(Y,X,X) &= \begin{bmatrix} \mathit{Attn}(y_1,X,X) \\ \vdots \\ \mathit{Attn}(y_{n_y},X,X) \end{bmatrix} = \begin{bmatrix} q(y_1) \\ \vdots \\ q(y_{n_y}) \end{bmatrix} \begin{bmatrix} k_1(x_1) & \dots & k_1(x_{n_x}) \\ \vdots & \ddots & \vdots \\ k_{d_q}(x_1) & \dots & k_{d_q}(x_{n_x}) \end{bmatrix} \begin{bmatrix} v(x_1) \\ \vdots \\ v(x_{n_x}) \end{bmatrix} \\
&= \begin{bmatrix} q(y_1)\cdot k(x_1) & \dots & q(y_1)\cdot k(x_{n_x}) \\ \vdots & \ddots & \vdots \\ q(y_{n_y})\cdot k(x_1) & \dots & q(y_{n_y})\cdot k(x_{n_x}) \end{bmatrix} \begin{bmatrix} v(x_1) \\ \vdots \\ v(x_{n_x}) \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{n_x}(q(y_1)\cdot k(x_i))v(x_i) \\ \vdots \\ \sum_{i=1}^{n_x}(q(y_{n_y})\cdot k(x_i))v(x_i) \end{bmatrix} \\
&\approx \begin{bmatrix} \int_{\Omega_x}(q(y_1)\cdot k(x))v(x)dx \\ \vdots \\ \int_{\Omega_x}(q(y_{n_y})\cdot k(x))v(x)dx \end{bmatrix} = \int_{\Omega_x}(q(Y)\cdot k(x))v(x)dx.
\end{aligned} \tag{10}
$$

Here, the discretization of input is $\{x_1, ..., x_{n_x}\}$ (as key and value vectors), and it can be changed to the discretization of output $\{y_1, ..., y_{n_y}\}$ (as query vectors) with cardinality changed from n_x to n_y. Using this mechanism, we can detach the dependences of discretization formats of input and output from the processor, by encoding arbitrary discretization $\{x_1, ..., x_{n_x}\}$ to a fixed size (n_z) of learnable latent set vectors, and decoding the latent set vectors to output arbitrary discretization $\{y_1, ..., y_{n_y}\}$. The discretization number is varied as n_x (arbitrary) $\rightarrow n_z$ (fixed) $\rightarrow n_y$ (arbitrary).

B Attention blocks

Mesh-independent neural operator (IPOT) consists of two types of attention blocks, cross- and self-attention blocks, which implement the respective attention mechanisms. The attention blocks have the shared structures following the Transformer-style architectures [19, 17, 27], which takes two input arrays, a query input $Y \in \mathbb{R}^{n_y \times d_y}$ and a key-value input $X \in \mathbb{R}^{n_x \times d_x}$,

$$
\begin{aligned}
O &= Y + \mathit{Attn}(\mathit{LayerNorm}(Y), \mathit{LayerNorm}(X), \mathit{LayerNorm}(X)), \\
\mathit{Attention}(Y,X,X) &= O + \mathit{FF}(\mathit{LayerNorm}(O)),
\end{aligned} \tag{11}
$$

where *LayerNorm* is layer normalization [38], *FF* consists of two point-wise feedforward neural networks with a GELU nonlinearity [57], and the exact calculation of attention *Attn* is

$$
\mathit{Attn}(X^q, X^k, X^v) = \mathit{Softmax}\left(\frac{QK^T}{\sqrt{d_q}}\right)V, \tag{12}
$$

where $Q = X^qW^q \in \mathbb{R}^{n_y \times d_q}$, $K = X^kW^k \in \mathbb{R}^{n_x \times d_q}$, and $V = X^vW^v \in \mathbb{R}^{n_x \times d_v}$ for a single headed attention. In the case of multi-headed attention, several outputs from different learnable parameters are concatenated and projected with the linear transformation.

C Datasets

In this section, we present an overview of the datasets and describe the corresponding equations and tasks along with the details of inputs and outputs. Table 6 presents a summary of the datasets for easy reference across different domains, including problems of PDEs solved on regular grids, irregular grids, and real-world data. The footnotes of each problem set are the corresponding URL address for the datasets, which are publicly available. The train/test split setting are also directly taken from respective benchmark datasets.

Table 6: Overview of datasets and the corresponding tasks in terms of size of input and output.

Dataset	Problem	Input	Output	Input size	Output size
Burgers	1D regular grid	$u(\cdot,0)$	$u(\cdot,1)$	1,024×1	1024×1
Darcy flow	2D regular grid	a	u	7,225×1	7,225×1
Navier-Stokes, ν=1e–3	2D regular grid	$w(\cdot,t)\|_{t\in[0,10]}$	$w(\cdot,t)\|_{t\in(10,50]}$	4,096×10	4,096×1×40
Navier-Stokes, ν=1e–4	2D regular grid	$w(\cdot,t)\|_{t\in[0,10]}$	$w(\cdot,t)\|_{t\in(10,30]}$	4,096×10	4,096×1×20
Navier-Stokes, ν=1e–5	2D regular grid	$w(\cdot,t)\|_{t\in[0,10]}$	$w(\cdot,t)\|_{t\in(10,20]}$	4,096×10	4,096×1×10
Airfoil	Transonic flow	Mesh point	Velocity	11,271×2	11,271×1
Elasticity	Hyper-elastic material	Point cloud	Stress	972×2	972×1
Plasticity	Plastic forging	Boundary condition	Displacement	62,620×1	62,620×4
Spherical shallow water	Spherical manifold	$u(\cdot,0)$	$u(\cdot,t)\|_{t\in(0,20]}$	8,192×2	8,192×2×19
ERA5	[Train] T_{2m} forecasting	$u(\cdot,t)\|_{t\in[-6,0]}$	$u(\cdot,t)\|_{t\in[1,7]}$	16,200×7	16,200×1×7
ERA5	[Test] Masked land	$u(\cdot,t)\|_{t\in[-6,0]}$	$u(\cdot,t)\|_{t\in[1,7]}$	11,105×7	16,200×1×7
ERA5	[Test] Masked sea	$u(\cdot,t)\|_{t\in[-6,0]}$	$u(\cdot,t)\|_{t\in[1,7]}$	5,095×7	16,200×1×7

C.1 Problems of PDEs solved on regular grids

Burgers' equation[1]**.** Burgers' equation is a non-linear parabolic PDE combining the terms of convection and diffusion. The benchmark problem of 1D Burgers' equation with periodic boundary conditions is defined as

$$\begin{aligned} \partial_t u(x,t) + \partial_x(u^2(x,t)/2) &= \nu\partial_{xx}u(x,t), && x\in(0,1), t\in(0,1] \\ u(x,0) &= u_0(x), && x\in(0,1) \end{aligned}$$

where $u_0 \sim \mu$ is the initial state generated from $\mu = \mathcal{N}(0, 625(-\Delta + 25I)^{-2})$ and $\nu = 0.1$ is the viscosity coefficient. The goal of operator learning is to learn mapping the initial state to the solution at $t = 1, \mathcal{G} : u(\cdot, 0) \mapsto u(\cdot, 1)$. During training, the input and output discretizations are given at 1,024 equispaced grids. To assess the performance of the trained models, we evaluate them on different discretizations during testing, specifically 512, 2,048, and 8,192 (refer to Table 1).

Darcy flow[1]**.** Darcy flow is a second-order elliptic PDE describing the flow of fluid through a porous medium. The benchmark problem of 2D steady-state Darcy flow on unit cell is defined as

$$\begin{aligned} -\nabla\cdot(a(x)\nabla u(x)) &= f(x), && x\in(0,1)^2 \\ u(x) &= 0, && x\in\partial(0,1)^2 \end{aligned}$$

where u is density of the fluid, $a \sim \mu$ is the diffusion field generated from $\mu = \mathcal{N}(0, (-\Delta + 9I)^{-2})$ with fixed forcing function $f = 1$. The goal of operator learning is to learn mapping the diffusion field to the solution of the density, $\mathcal{G} : a \mapsto u$. During training, the discretizations of input and output are sampled at 85×85=7,225 regular grids. To assess the performance of the trained models, we evaluate them on different discretizations during testing, specifically 141×141 and 211×211 regular grids (refer to Table 1).

Navier-Stokes equation[1]**.** Navier-Stokes equation describes the dynamics of a viscous, incompressible fluid. The benchmark problem of the 2D Navier-Stokes equation in vorticity form on the unit torus is defined as

$$\begin{aligned} \partial_t w(x,t) + u(x,t)\cdot\nabla w(x,t) &= \nu\Delta w(x,t) + f(x), && x\in(0,l)^2, t\in(0,T] \\ \nabla\cdot u(x,t) &= 0, && x\in(0,l)^2, t\in[0,T] \\ w(x,0) &= w_0(x), && x\in(0,l)^2 \end{aligned}$$

where u is the velocity field, $w = \nabla \times u$ is the vorticity field, $w_0 \sim \mu$ denotes the initial vorticity field generated from $\mu = \mathcal{N}(0, 7^{3/2}(-\Delta + 49I)^{-2.5})$ with periodic boundary conditions, and the forcing function is kept $f(x) = 0.1\,(\sin(2\pi\,(x_1 + x_2)) + \cos(2\pi\,(x_1 + x_2)))$. The goal of operator learning is to learn the mapping from the initial times $t \in [0, 10]$ of the vorticity fields to further trajectories up to $t = T$, defined by $w(\cdot,t)|_{t\in[0,10]} \mapsto w(\cdot,t)|_{t\in(10,T]}$. The time total duration of each trajectory is T=50, 30, and 10 corresponding to the viscosity coefficients ν=1e–3, 1e–4, and 1e–5, respectively, for each dataset [8]. 1,000 instances are used for training and 100 for testing. During both training and testing, the spatial discretizations of input and output are sampled at 65×65=4,096 regular grids (refer to Table 1).

[1]`https://github.com/neuraloperator/neuraloperator`

C.2 Problems of PDEs solved on irregular grids

Airfoil[2]**.** The benchmark problem of the airfoil is based on the transonic flow over an airfoil, which is governed by Euler's equation. The equation is defined as

$$\partial_t \rho + \nabla \cdot (\rho^f v) = 0, \qquad \partial_t(\rho u) + \nabla \cdot (\rho u \otimes u + pI) = 0, \qquad \partial_t E + \nabla \cdot \big((E+p)u\big) = 0, \tag{13}$$

where ρ is the density, u is the velocity field, p is the pressure, and E is the total energy. The far-field boundary conditions are $\rho_\infty = 1, p_\infty = 1$, and Mach number $M_\infty = 0.8$, with no penetration imposed at the airfoil. The shape and mesh grids of the datasets used in this study are adopted from [11], which consists of variations of the NACA-0012 airfoils. The dataset is split into 1,000 instances for training and 200 instances for testing. The inputs and outputs are given as the mesh grids and the corresponding velocities in terms of Mach number.

Elasticity[2]**.** The benchmark problem of elasticity involves hyper-elastic material, which is governed by constitutive equations. The equation on the bounded unit cell $\Omega \in [0,1]^2$ is defined as

$$\rho \frac{\partial^2 u}{\partial t^2} + \nabla \cdot \sigma = 0, \tag{14}$$

where ρ is mass density, u is the displacement vector, and σ is the stress tensor, respectively. The random-shaped cavity at the center has a radius that is sampled as $r = 0.2 + \frac{0.2}{1+\exp(\tilde{r})}$, where $\tilde{r}$ is drawn from a distribution $\tilde{r} \sim \mathcal{N}(0, 4^2(-\nabla + 3^2)^{-1})$. The unit cell is fixed on the bottom edges and a tension traction of $t = [0, 100]$ is applied on the top edge. The hyper-elastic material used is the incompressible Rivlin-Saunders material. The dataset and split setting are also directly taken from [11], where 1,000 instances are for training and 200 for testing. The inputs and outputs are given as coordinates of point clouds and the corresponding stress.

Plasticity[2]**.** The plasticity benchmark problem involves 3D plastic forging, where the governing constitutive equation is the same as Equation 14 over time. The objective is to learn a mapping from the initial boundary condition to the mesh grids and displacement vectors over time. The target solution has the dimension of 62,620×4, where the output queries are structured on a 101×31 mesh grid over 20 time steps. This results in a total of 62,620 output queries, each consisting of 2 mesh grids and 2 displacement vectors, thus yielding 4 dimensions. The dataset and split setting are also directly taken from [11], where 900 instances are for training and 80 for testing.

Spherical shallow water[3]**.** We consider another PDE problem of the 3D spherical shallow water equation [58, 41]. The shallow water equation is

$$\partial_t u = -fk \times u - g\nabla h + \nu \Delta u, \qquad \partial_t h = -h\nabla \cdot u + \nu \Delta h, \tag{15}$$

where k is the unit normal vector to the spherical surface, u is the velocity field tangent to the spherical surface, $w = \nabla \times u$ is the vorticity field, and h is the thickness of the sphere. The observational data at time t is given as $v_t = (w_t, h_t)$. f, g, ν, Ω are the parameters of the Earth, which can be found in [58]. We follow [41] to create symmetric phenomena on the northern and southern hemispheres for the initial conditions. $u_0(\phi, \theta)$ is the initial zonal velocity written as:

$$u_0(\phi,\theta) = \begin{cases} \left(\frac{u_{max}}{e_n} \exp\left(\frac{1}{(\phi-\phi_0)(\phi-\phi_1)}\right), 0\right) & \text{if } \phi \in (\phi_0, \phi_1), \\ \left(\frac{u_{max}}{e_n} \exp\left(\frac{1}{(\phi-\phi_0)(\phi-\phi_1)}\right), 0\right) & \text{if } \phi \in (-\phi_1, -\phi_0), \\ (0,0) & \text{otherwise.} \end{cases}$$

where latitude and longitude $\phi, \theta \in \left[-\frac{\pi}{2}, \frac{\pi}{2}\right] \times [-\pi, \pi]$, $u_{max} \sim \mathcal{U}(60, 80)$ is the maximum velocity sampled from uniform distribution range from 60 to 80, $\phi_0 = \frac{\pi}{7}$, $\phi_1 = \frac{\pi}{2} - \phi_0$, and $e_n = \exp\left(-\frac{4}{(\phi_1-\phi_0)^2}\right)$. We also follow [41] to initialize the perturbed water height $h_0(\phi,\theta)$ from the original one:

$$h_0(\phi,\theta) = \hat{h}\cos\phi \exp\left(-\left(\frac{\theta}{\alpha}\right)^2\right)\left[\exp\left(-\left(\frac{\phi_2-\phi}{\beta}\right)^2\right) + \exp\left(-\left(\frac{\phi_2+\phi}{\beta}\right)^2\right)\right]$$

where $\phi_2 = \frac{\pi}{4}$, $\hat{h} = 120m$, $\alpha = \frac{1}{3}$, $\beta = \frac{1}{15}$ are constants from [58].

[2]`https://github.com/neuraloperator/Geo-FNO`

[3]`https://github.com/mkirchmeyer/DINo`

C.3 Problem of real-world data

ERA5 reanalysis[4]**.** The ERA5 reanalysis database, provided by the European Centre for Medium-Range Weather Forecasts (ECMWF) [42], offers extensive hourly and monthly measurements for various parameters, including temperature, precipitation, and wind speed at different pressure levels. For our experiments, we downloaded a subset of the ERA5 database, specifically for the 2m temperatures at 00:00 UTC from January 1st, 2018 to March 31st, 2023. We divided our dataset into non-overlapping segments of 7 consecutive days. The goal is to map the temperature field of the previous 7 days to the temperature field of the next 7 days, $u(\cdot,t)|_{t\in[-6,0]} \mapsto u(\cdot,t)|_{t\in[1,7]}$. The dataset comprised a total of 275 input-output pairs, where 250 instances were used for training and 25 instances were used for testing. We downsampled each frame by a factor of 8, resulting in a resolution of 2° (90×180) for training, and we tested the pre-trained models on various resolutions, ranging from 0.25° to 4°, as well as on masked input scenarios (as depicted in Figure 3). Masked input with a land mask[5] [50] leads to 11,105 points, while the sea region contains 5,095 points.

D Implementation details

D.1 Architecture details

Table 7: Architecture details of IPOT.

Dataset	Regular			Irregular				Real
Problem	Burgers	Darcy flow	Navier-Stokes	Airfoil	Elasticity	Plasticity	Shallow water	ERA5
				Positional encoding				
Frequency bins	64	[32, 32]	[12, 12]	[8, 8]	[16, 16]	[3, 3, 3]	[20, 20, 20]	[64, 64]
Max frequency	64	[32, 32]	[20, 20]	[16, 16]	[16, 16]	[12, 12, 12]	[32, 32, 32]	[64, 128]
Positional encoding	1,024×129	7,225×130	4,096×50	11,271×34	972×132	62,620×21	8,192×123	16,200×258
				Encoder				
Input function values	1,024×1	7,225×1	4,096×1	11,271×1	972×1	62,620×1	8,192×1	16,200×1
Latent channels	64	64	128	64	64	64	128	128
Number of heads	8	1	1	1	1	1	2	1
Inputs	1,024×130	7,225×131	4,096×51	11,271×35	972×133	62,620×22	8,192×124	16,200×259
				Processor				
Learnable queries	256×64	256×64	512×128	128×64	512×64	256×64	256×128	512×128
Latent channels	64	64	128	64	64	64	128	128
Number of heads	8	8	4	4	4	2	8	8
Number of blocks	1	4	2	2	4	2	2	4
				Decoder				
Output queries	1,024×1	7,225×2	4,096×2	11,271×2	972×4	62,620×3	8,192×3	16,200×2
Latent channels	64	64	128	64	64	64	128	128
Number of heads	8	1	1	1	1	1	2	1
Outputs	1,024×1	7,225×1	4,096×1	11,271×1	972×1	62,620×4	8,192×2	16,200×1

D.2 Training details

Table 8: Training details for IPOT.

Data type	Regular			Irregular				Real
Problem	Burgers	Darcy flow	Navier-Stokes	Airfoil	Elasticity	Plasticity	Shallow water	ERA5
Batch size	20	10	100	20	10	10	64	10
Epochs	2,000	1,000	5,000	1,600	1,600	1,600	3,000	5,000
Learning rate	0.001	0.001	0.001	0.001	0.001	0.001	0.001	0.001
Learning rate decay	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
Step decay	250	200	250	200	200	200	500	500

Training. The experiments are conducted on a 24GB NVIDIA GeForce RTX 3090 GPU and use AdamW optimizer [59] with an initial learning rate of 1e–3. The implemented architectures and corresponding training hyperparameters for each problem are summarized in Table 7 and Table 8, respectively.

Evaluation metric. We use relative L_2 error for the objective functions and evaluation metrics for test errors, where we follow the convention of related literature [5, 8]. The relative L_2 error is defined as $E_{a\sim\mu}[\mathcal{L}(\mathcal{G}_\theta(a),u)] = \frac{1}{N}\sum_{i=1}^{N}\frac{\|u-\mathcal{G}_\theta(a)\|_2}{\|u_i\|_2}$ where N is the dataset size of input-output pairs $(a_i,u_i)_{i=1}^{N}$.

[4] `https://www.ecmwf.int/en/forecasts/datasets/browse-reanalysis-datasets`

[5] `https://downloads.psl.noaa.gov/Datasets/noaa.oisst.v2/lsmask.nc`

D.3 Baselines

The results for the problems on regular grids and irregular grids were obtained from the related literature, including DeepONet [2], GNO [6], FNO [8], FT/GT [16], and OFormer [40] for experiments on regular grids, Geo-FNO [11], MPPDE [14], and DINO [41] for experiments on irregular grids. For the evaluation of baselines on real data (ERA5), we produced the results using their original codes of FNO[6] and OFormer[7]. Since the problem is formulated as a time-stepping system similar to the Navier-Stokes equation, we adopted almost the same architectures that were used in the Navier-Stokes problem for the temperature forecasting task. However, due to the requirement of having the input and output share the same regular grid structure in FNO [5, 8], incorporating interpolated values on the masked region is necessary for masked input tasks. To adopt FNO to masked inputs, we use cubic interpolation methods to obtain interpolated input values for land or sea coordinates. These interpolated values are then combined with the masked inputs, as shown in Figure 4. However, due to the masked regions being irregular and complex, the interpolated input values may not accurately represent the real temperature fields. As a result, this leads to significant performance degradation for FNO in forecasting the output temperature fields. On the other hand, IPOT does not require such problem-specific pre-processing.

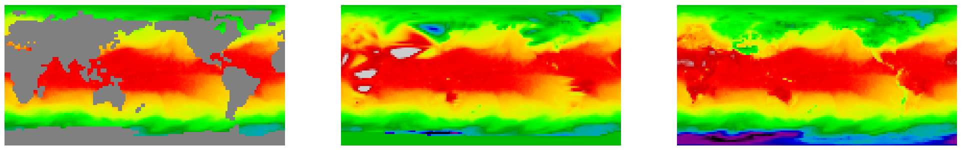

Figure 4: Land regions are masked which is motivated by situations when observations are only available for the sea surface (left). In order to adopt FNO to masked input tasks, we incorporated interpolated values (middle). However, there can be a large gap between the real temperature field (right) and the interpolated inputs.

E Additional Results

E.1 Number of inducing points

We conducted experiments on ERA5 data with varying the number of latent query vectors, ranging from 32 to 512, to evaluate the effect of the number of inducing points, as shown in Table 9. The results demonstrate that increasing the number of latent query vectors improves the performance of IPOT. Notably, when the number of inducing points is 256, IPOT sufficiently outperforms other baselines. However, when the number of inducing points is too small, IPOT exhibits poor performance compared to other baselines (see Table 4).

Table 9: Performance of IPOT with varying number of inducing points

Number of inducing points, n_z	32	64	128	256	512
Error	3.07e–2	1.45e–2	1.30e–2	6.87e–2	6.44e–3

E.2 Computational complexity on Different resolutions

We compared the computational complexity of different models on the ERA5 data at different resolutions, as shown in Figure 5. The time complexities were assessed by measuring the inference time in seconds for processing observational data during testing, and the memory complexities were evaluated by recording CUDA memory allocation. It is observed that IPOT without latent inducing points, which serves as a surrogate for a standard Transformer with quadratic complexity in self-attention, does not scale well in terms of both time and memory costs. OFormer, based on the Galerkin Transformer, exhibits sub-quadratic complexity compared to the standard Transformer but poses memory limitations when the size of n reaches 10^6. In contrast, FNO and IPOT with 256 or

[6]`https://github.com/neuraloperator/neuraloperator`

[7]`https://github.com/BaratiLab/OFormer`

512 inducing points scale effectively up to $n = 10^6$, making them suitable for handling large-scale observational data due to their efficiency in both time and memory complexity.

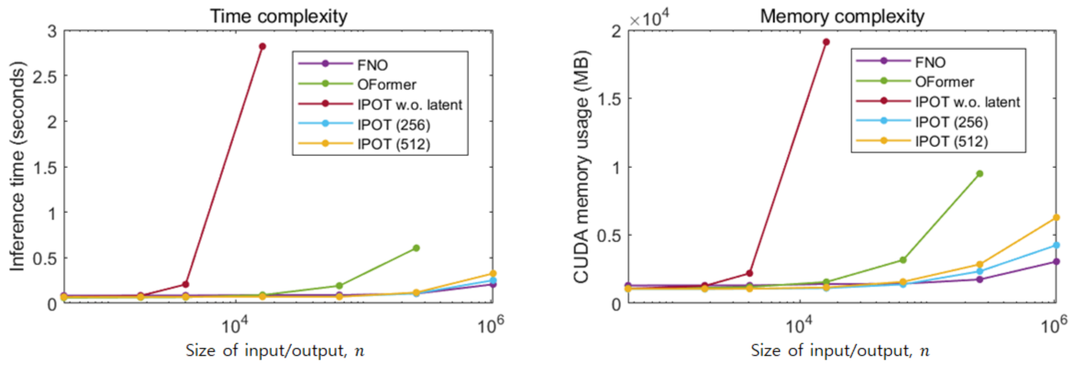

Figure 5: Complexity comparisons on different resolutions. We compare the different models in terms of inference time (left) and CUDA memory usage (right) with different sizes of input/output.

E.3 Visualizations of daily temperature field.

Figure 6 provides the visualizations of daily temperature fields at 2m above the surface. The visualizations display the ground truth data as well as the predictions from IPOT, FNO, and OFormer, respectively.

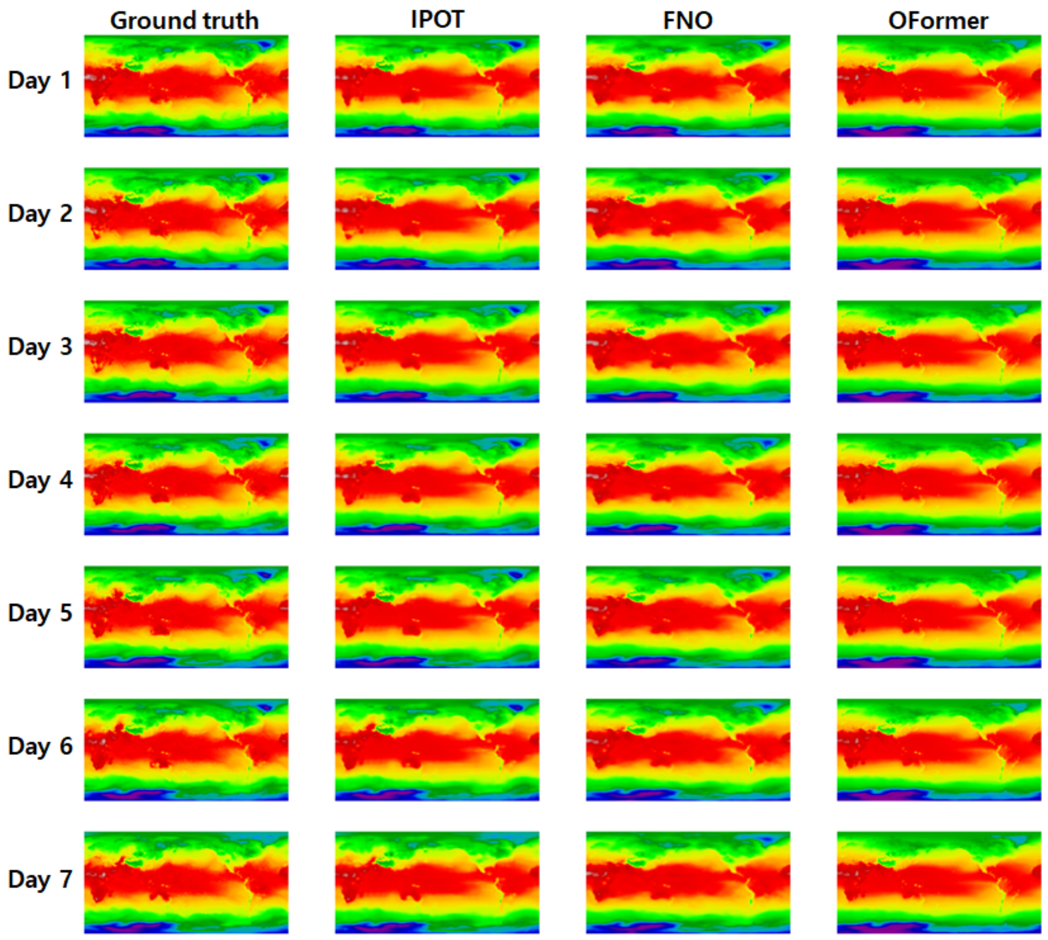

Figure 6: We visualize the ground truth and the predictions of daily temperatures. The predictions are computed by IPOT, FNO, and OFormer, respectively.