OPERATOR LEARNING ON FREE-FORM GEOMETRIES

Anonymous authors
Paper under double-blind review

ABSTRACT

Operator Learning models usually rely on a fixed sampling scheme for training which might limit their ability to generalize to new situations. We present CORAL, a new method which leverages **C**oordinate-Based Networks for **O**pe**RA**tor **L**earning without any constraints on the training mesh or input sampling. CORAL is able to solve complex Initial Value Problems such as 2D Navier-Stokes or 3D-spherical Shallow-Water and can perform zero-shot super-resolution to recover a dense grid, even when the training grid is irregular and sparse. It can also be applied to the task of geometric design with structured or point-cloud data, to infer the steady physical state of a system given the characteristics of the domain.

1 INTRODUCTION

Most methods for solving physical problems rely on architectures with spatial inductive bias, such as CNNs (Ayed et al. (2020), Bézenac et al. (2019)) or GNNs (Pfaff et al. (2021) Brandstetter et al. (2022)), and therefore expect an input over a mesh. Similarly recent approaches such as Neural Operators transform the physical state with both a local map and a kernel integrator requiring a mesh over the domain (Kovachki et al., 2021). Moreover, the state of the art Fourier Neural Operator (Li et al., 2021) approximates this integrator with a Fourier transform and requires a regular rectangular grid to apply the Fast Fourier Transform (FFT). Geo-FNO (Li et al., 2022) is a first attempt to mitigate the FNO model dependency to fixed grids. DeepONet and its variants (Lu et al., 2022) have mesh requirements over the inputs as they expect the same sensors (same number, same locations, same order) for a new input and cannot adapt to a change of mesh (Prasthofer et al., 2022).

Therefore, existing methods for operator learning may face difficulties when dealing with applications, including variable sampling, i.e. missing inputs or a change of grid between train and test, different geometries, i.e. a change of domain or boundary conditions between the train and test, or data lying on non-euclidean manifolds, for instance 2D-Torus or 3D-Sphere.

Implicit Neural Representation (INRs) (Park et al., 2019), (Mildenhall et al., 2020) and Physics Informed Neural Networks (PINNs) (Raissi et al., 2019) only depend on a coordinate system over a spatial domain and can off-the-shelf be applied with unstructured data, missing inputs or complex geometries. Initially aimed at representing a single function at a time, they can be combined with a hypernetwork (Ha et al., 2017), (Sitzmann et al., 2020b) to work with different samples.

In this work, we aim at learning a PDE-based Operator between function spaces with a model that can easily overcome the aforementioned limitations. We introduce CORAL, an INR-based model for Operator Learning, with the following contributions. In order to be **discretization-invariant**, CORAL encodes the input and output functions into small latent spaces with the help of INRs, and infers the mapping between them with an MLP. To facilitate this inference task, we learn to **quickly encode** physical data on a given mesh with a meta-learning training procedure. Unlike most existing methods, our framework is able to learn an Operator conditioned on an irregular mesh while maintaining similar performance when applied to a **new grid unseen at training time**. We **empirically validate** the performance of our framework vs state-of-the-art models on a selection of Initial Value Problems (IVP) and Geometric Design Tasks (GD).

2 PROBLEM DESCRIPTION

Problem setting. Let $\Omega \subset \mathbb{R}^d$ be a bounded open set of spatial coordinates. We seek to learn a mapping $\mathcal{G}$ from one infinite-dimensional space $\mathcal{A} \subset L^2(\Omega, \mathbb{R}^{d_a})$ to another $\mathcal{U} \subset L^2(\Omega, \mathbb{R}^{d_u})$

through an i.i.d collection of pointwise evaluations of input-output functions. At training time, we have access to n pairs of input and output functions $(a_i, u_i)_{i=1}^n$ evaluated over a free-form spatial grid $\mathcal{X}_{tr}$. To simplify the notation, we note $a|_{\mathcal{X}_{tr}} = (a(x))_{x\in\mathcal{X}_{tr}}$ and $u|_{\mathcal{X}_{tr}} = (u(x))_{x\in\mathcal{X}_{tr}}$ the vectors of the function values over the training grid. At test time, i.e. when a new input function a is considered, we evaluate the model on a spatial grid $\mathcal{X}_{te}$ that can be different from $\mathcal{X}_{tr}$. In the context of PDEs, we target two different tasks • solving an **Initial Value Problem**, i.e. mapping the initial condition $u_0 \doteq x \mapsto u(x, t = t_0)$ to the solution at a certain time $u_T \doteq x \mapsto u(x, t = T)$, • or solving a **Geometric Design** task, i.e. mapping a parameter function a characterizing an object geometry associated with a meshed description of the domain (e.g. an airfoil) to the equation solution u of the quantity of interest (e.g. its Mach number).

3 MODEL

Our model mitigates the dependency on fixed grids for operator learning by learning a mapping between the parameter space of two implicit neural representations. At inference stage, we encode the input function a in a small latent code z_a with a spatial encoder $e_a : \mathcal{A} \mapsto \mathbb{R}^{d_z}$. Then we infer the output latent code with an MLP $g_\psi : \mathbb{R}^{d_z} \mapsto \mathbb{R}^{d_z}$ and finally decode it to a spatial function with a decoder $\xi_u : \mathbb{R}^{d_z} \mapsto \mathcal{U}$. Overall our operator can be written as $\tilde{\mathcal{G}} = \xi_u \circ g_\psi \circ e_a$ and we represent its flow in Figure 1. During training, we learn to reconstruct the input and output functions a and u with an encoder-decoder framework. This requires an input decoder $\xi_a : \mathbb{R}^{d_z} \mapsto \mathcal{A}$ and an output encoder $e_u : \mathcal{U} \mapsto \mathbb{R}^{d_z}$, even though they are not used during inference.

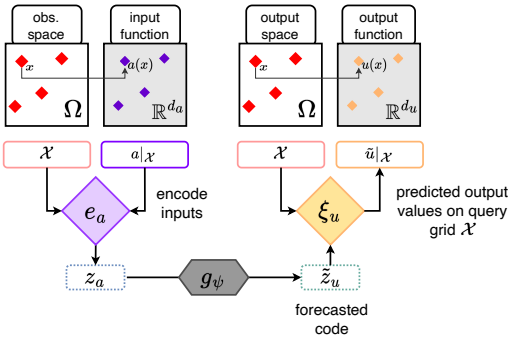


Figure 1: Inference for CORAL. First, the model embeds the input function a without constraints on the locations of the observed sensors into a latent code z_a, then infers the latent $\tilde{z}_u$ and finally predicts the output value $\tilde{u}(x)$ for any query coordinate $x \in \Omega$. For the grid $\mathcal{X}$, we use the vector notation $a|_{\mathcal{X}} = (a(x))_{x\in\mathcal{X}}$, $\tilde{u}|_{\mathcal{X}} = (\tilde{u}(x))_{x\in\mathcal{X}}$.

3.1 MODEL ARCHITECTURE

The model consists of three main components.

Decoder Let $f_{\theta_u} \in \mathcal{U}$ be a parameterized INR with paramaters θ_u. We derive a space-continuous decoder by conditioning the network f_{θ_u} with a latent code z_u. The corresponding function is denoted ξ_u and defined as $\xi_u : z_u \mapsto f_{\theta_u, h_u(z_u)}$ where h_u is a hypernetwork to be later defined. $f_{\theta_u, h_u(z_u)}$ is an INR, i.e. a function of spatial coordinates, which we can freely query at any point within the domain. We thus have $\forall x \in \Omega, \tilde{u}(x) = \xi_u(z_u)(x) = f_{\theta_u, h_u(z_u)}(x)$. Similarly for the input, by noting $f_{\theta_a} \in \mathcal{A}$ a parameterized INR with parameters θ_a, we have $\forall x \in \Omega, \tilde{a}(x) = \xi_a(z_a)(x) = f_{\theta_a, h_a(z_a)}(x)$. We denote w_u and w_a the parameters of h_u and h_a. See Figure 6 and 7 in Appendix A.4 for details.

Encoder Given a function a, the encoder outputs a code $z_a = e_a(a)$ such that the decoder can retrieve the function a, i.e. $\forall x \in \mathcal{X}, \xi_a(z_a)(x) = a(x)$. To find an approximate solution to this inverse problem, our encoder maps the function a to the final code $e_a(a) = z_a^{(K)}$ of a gradient descent

optimization:

$$\begin{aligned} &z_a^{(0)} = 0; \\ &z_a^{(k)} = z_a^{(k-1)} - \alpha \nabla_{z_a^{(k-1)}} \mathcal{L}_\mu(\xi_a(z_a^{(k-1)}), a); \text{for } 1 \leq k \leq K; \end{aligned} \tag{1}$$

where α is the inner learning rate, K the number of inner steps, $\mathcal{L}_\mu(v, w) = \mathbb{E}_{x\sim\mu}[(v(x) - w(x))^2]$ for a measure $\mu \in \sigma(\Omega)$. We define in the same way e_u the encoder for the output functions. Note that in practice, the measure μ is defined through the observation grid $\mathcal{X}$, $\mu(\cdot) = \mu|_{\mathcal{X}}(\cdot) = \sum_{x\in\mathcal{X}} \delta_x(\cdot)$ where $\delta_x(\cdot)$ is the dirac measure. Since we can query the INRs anywhere within the domain, we can hence freely encode functions without mesh constraints. This is the essential part of the architecture that enables us to feed data defined on different grids to the model. We show the encoding flow in Appendix A.4 Figure 8.

Inference model We infer the latent code $\tilde{z}_u = g_\psi(z_a)$ with a feed-forward Neural Network with parameters ψ.

3.2 Modulated SIREN

We choose SIREN (Sitzmann et al., 2020b) – a state-of-the-art coordinate-based network – as the standard INR block of our architecture. SIREN is a neural network that uses sine activations:

$$f_\theta(x) = \boldsymbol{W}_L(\Phi_L \circ \Phi_{L-1} \circ \Phi_{L-2} \circ ....\Phi_0(x)) + \boldsymbol{b}_L, \text{with } \Phi_i(h_i) = \sin(\omega_0(\boldsymbol{W}_i h_i + \boldsymbol{b}_i)) \tag{2}$$

where $w_0 \in \mathbb{R}_+^*$ is a frequency hyperparameter. Instead of generating all the SIREN parameters $(\boldsymbol{W}_i, \boldsymbol{b}_i)_{i=0}^L$ with the hypernetwork h, we implement scale and shift modulations (Perez et al., 2018). Formally, given an input code z the hypernetwork yields the modified set of parameters $(\theta, h(z)) = (\boldsymbol{\gamma}_i \odot \boldsymbol{W}_i, \boldsymbol{b}_i + \omega_0 \boldsymbol{W}_i \boldsymbol{\beta}_i)_{i=0}^L$, where $\boldsymbol{\gamma}_i = \boldsymbol{A}_i z + \boldsymbol{c}_i$ and $\boldsymbol{\beta}_i = \boldsymbol{B}_i z + \boldsymbol{d}_i$.

3.3 Training

We implement a two-step training procedure that first adjusts the modulated INR parameters, before training the inference model. We outline the training pipeline in Appendix A.4 Figure 9. Formally, the optimization problem is the following.

$$\begin{aligned} &\arg\min_\psi \mathbb{E}_{a,u\sim\nu_a,\nu_u} \text{MSE}(g_\psi(\tilde{e}_a(a)), \tilde{e}_u(u)) \\ \text{s.t. } &\tilde{e}_a = \arg\min_{\xi_a, e_a} \mathbb{E}_{a\sim\nu_a} \mathcal{L}(\xi_a \circ e_a(a), a) \\ \text{and } &\tilde{e}_u = \arg\min_{\xi_u, e_u} \mathbb{E}_{u\sim\nu_u} \mathcal{L}(\xi_u \circ e_u(u), u) \end{aligned} \tag{3}$$

In our architecture, note that the functions (e_u, e_a, ξ_a, ξ_u) are parameterized by the weights $(\theta_a, w_a, \theta_u, w_u)$ of the INRs and hypernetworks, and thus optimization is tackled on the latter parameters. During training, we constrain the encoder to take only a few steps of gradient descent to facilitate the inference task. We found in practice that a meta-learning strategy based on CAVIA (Zintgraf et al., 2019) was able to faithfully reconstruct new physical data in a few steps. In other words, at each epoch we encode a batch $\mathcal{B}$ of functions $(a_i)_{i\in\mathcal{B}}, (u_i)_{i\in\mathcal{B}}$ into their respective codes $(z_{a_i})_{i\in\mathcal{B}}, (z_{u_i})_{i\in\mathcal{B}}$ following Equation 1 with K inner steps, and update the common parameters during the outer loop. This is a second-order meta-learning algorithm as the gradient of the outer loop backpropagates through the K inner steps of encoding. See Appendix A.4 Algorithm 1 for details.

4 Experiments

We performed experiments on two tasks, solving an initial value problem (Section 4.1 and Appendix A.2) and solving a design problem (Section 4.2). All metrics are expressed in relative L2 error.

4.1 Initial Value Problem

• **Datasets** We consider a dataset from the following PDE: **2D-Navier-Stokes equation** (*Navier-Stokes*) for a viscous, incompressible fluid in vorticity form on the unit torus: $\frac{\partial w}{\partial t} + u \cdot \nabla w = \nu \Delta w + f$,

$\nabla u = 0$ for $x \in \Omega, t > 0$, where $\nu = 10^{-3}$ is the viscosity coefficient. We learn to map the vorticity at time $t_0 = 10$, i.e. $w(\cdot, t_0 = 10)$ to the vorticity at time $T = 20$, i.e $w(\cdot, T = 20)$ (more details in Appendix A.3). • **Evaluation**. **(Same-grid)** First, we validate that the model is able to generalize to a new test input with the same sensors as for training : $\mathcal{X}_{te} = \mathcal{X}_{tr}$. **(Regular up-sampling)** Then, we consider a more challenging setting where the train grid is a regular subsampling of the test grid, and wish to perform well on this up-sampling task for new inputs. **(Irregular up-sampling)** Similarly, we assess the up-sampling capability of the model when the train grid has been obtained by randomly selecting π_{tr} percent of the test grid sensors irregularly sampled. The train size is $n = 1000$ samples and test size is 200 samples. • **Baselines** We compare our model to popular Operator Learning methods: **DeepONet** (Lu et al., 2021) an Operator Network that expects fixed inputs. **FNO** (Li et al., 2021) a Neural Operator that can work with regular grids of different resolutions. **Geo-FNO** (Li et al., 2022) has been designed for irregular meshes. • **Results** In Table 1a, we can see that CORAL outperforms DeepONet and is competitive with FNO for *Navier-Stokes* on regular grids. Furthermore, we observe in Table 1b that it outperforms FNO in all up-sampling tasks, with more stable results across train and test resolutions. Besides, we show in Appendix A.2 Table 3 that CORAL is able to learn on an irregular grid and generalize to a new dense grid, with performance similar to the regular up-sampling setting.

Table 1: Test results on *Navier-Stokes*.

(a) **Same-grid** - $\mathcal{X}_{tr} = \mathcal{X}_{te}$

Networks	64×64	128×128
FNO	0.00378	0.00339
DeepONet	0.0398	0.0384
CORAL	**0.00259**	**0.00209**

(b) **Regular up-sampling**

$\mathcal{X}_{tr} \downarrow$	$\mathcal{X}_{te} \rightarrow$	128×128	256×256
64×64	FNO	0.1153	0.1321
	CORAL	**0.00267**	**0.0231**
128×128	FNO	0.00339	0.0648
	CORAL	**0.00209**	**0.0225**

4.2 Geometric Design

• **Euler equation** (*Airfoil*) We consider the transonic flow over an airfoil, where the governing equation is the Euler equation described in A.3. Each sample i represents a different airfoil shape, and thus has a different fluid domain Ω_{a_i} and a different mesh $\mathcal{X}_{a_i}$. The mesh is obtained by deforming a rectangular grid $\mathcal{X} \subset \Omega$ to obtain more nodes near the airfoil. The input function a is the grid deformation, mapping coordinates $\xi \in \Omega$ to physical coordinates $a(\xi) = x \in \Omega_a$, and the output function represents the mach numbers.

• **Evaluation** At test time, we assess the ability of the model to generalize for a new input geometry, e.g. a new airfoil shape. The train size is $n = 1000$ samples and test size is 200. • **Results** On *Airfoil*, CORAL achieves state-of-the-art results with the lowest relative error among all models (Table 2). From a qualitative point of view, we notice in Figure 2 that CORAL is able to predict with high accuracy the output values near the airfoil.

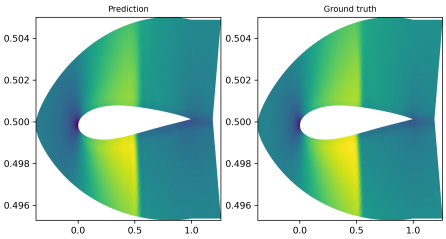


Figure 2: Qualitative results on *Airfoil*.

Table 2: Test results on *Airfoil*.

Networks	Rel. L2 err.
Geo-FNO	0.0138
FNO Interpolation	0.0421
UNet Interpolation	0.0519
CORAL	**0.0107**

5 Conclusion

We propose CORAL, a new approach for Operator Learning without any constraints on the training mesh or the input sampling. We assess the performance of CORAL on a couple of Initial Value Problems and its abilities to generalize to new initial conditions and to new grids. CORAL shows promising results for system design and could become an alternative choice of surrogate models.

REFERENCES

Ibrahim Ayed, Emmanuel De Bezenac, Arthur Pajot, and Patrick Gallinari. Learning the spatio-temporal dynamics of physical processes from partial observations. *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, 2020-May, 2020. ISSN 15206149. doi: 10.1109/ICASSP40776.2020.9053035.

Johannes Brandstetter, Daniel E Worrall, and Max Welling. Message passing neural pde solvers. *International Conference on Learning Representations*, 2022.

Keaton J. Burns, Geoffrey M. Vasil, Jeffrey S. Oishi, Daniel Lecoanet, and Benjamin P. Brown. Dedalus: A flexible framework for numerical simulations with spectral methods. *Physical Review Research*, 2, 2020. ISSN 26431564. doi: 10.1103/PhysRevResearch.2.023068.

Emmanuel De Bézenac, Arthur Pajot, and Patrick Gallinari. Deep learning for physical processes: incorporating prior scientific knowledge. *Journal of Statistical Mechanics: Theory and Experiment*, 2019, 12 2019. ISSN 17425468. doi: 10.1088/1742-5468/ab3195.

Peter Yichen Chen, Jinxu Xiang, Dong Heon Cho, Yue Chang, G A Pershing, Henrique Teles Maia, Maurizio Chiaramonte, Kevin Carlberg, and Eitan Grinspun. Crom: Continuous reduced-order modeling of pdes using implicit neural representations. *International Conference on Learning Representations*, 6 2022. URL http://arxiv.org/abs/2206.02607.

Zhiqin Chen and Hao Zhang. Learning implicit fields for generative shape modeling. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2019-June, 2019. ISSN 10636919. doi: 10.1109/CVPR.2019.00609.

Emilien Dupont, Hyunjik Kim, S. M. Ali Eslami, Danilo Rezende, and Dan Rosenbaum. From data to functa: Your data point is a function and you can treat it like one. *Proceedings of the 39 th International Conference on Machine Learning*, 1 2022. URL http://arxiv.org/abs/2201.12204.

Rizal Fathony, Anit Kumar Sahu, Devin Willmott, and J Zico Kolter. Multiplicative filter networks. *International Conference on Learning Representations.*, 2021.

David Ha, Andrew M. Dai, and Quoc V. Le. Hyper networks. *5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings*, 2017.

Nikola Kovachki, Zongyi Li, Burigede Liu, Kamyar Azizzadenesheli, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Neural operator: Learning maps between function spaces. 8 2021. URL http://arxiv.org/abs/2108.08481.

Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. *International Conference on Learning Representations.*, 10 2021. URL http://arxiv.org/abs/2010.08895.

Zongyi Li, Daniel Zhengyu Huang, Burigede Liu, and Anima Anandkumar. Fourier neural operator with learned deformations for pdes on general geometries. 7 2022. URL http://arxiv.org/abs/2207.05209.

David B. Lindell, Dave Van Veen, Jeong Joon Park, and Gordon Wetzstein. Bacon: Band-limited coordinate networks for multiscale scene representation. *Conference on Computer Vision and Pattern Recognition*, 12 2022. URL http://arxiv.org/abs/2112.04645.

Lu Lu, Pengzhan Jin, and George Em Karniadakis. Deeponet: Learning nonlinear operators for identifying differential equations based on the universal approximation theorem of operators. *Nat Mach Intell*, 3:218–229, 10 2021. doi: 10.1038/s42256-021-00302-5. URL http://arxiv.org/abs/1910.03193http://dx.doi.org/10.1038/s42256-021-00302-5.

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*, volume 393. Elsevier B.V., 4 2022. doi: 10.1016/j.cma.2022.114778.

Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2019-June, 2019. ISSN 10636919. doi: 10.1109/CVPR.2019.00459.

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. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 12346 LNCS, 2020. ISSN 16113349. doi: 10.1007/978-3-030-58452-8_24.

Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2019-June, 2019. ISSN 10636919. doi: 10.1109/CVPR.2019.00025.

Ethan Perez, Florian Strub, Harm De Vries, Vincent Dumoulin, and Aaron Courville. Film: Visual reasoning with a general conditioning layer. *32nd AAAI Conference on Artificial Intelligence, AAAI 2018*, 2018. ISSN 2159-5399. doi: 10.1609/aaai.v32i1.11671.

Tobias Pfaff, Meire Fortunato, Alvaro Sanchez-Gonzalez, and Peter W. Battaglia. Learning mesh-based simulation with graph networks. *International Conference on Learning Representations.*, 10 2021. URL http://arxiv.org/abs/2010.03409.

Michael Prasthofer, Tim De Ryck, and Siddhartha Mishra. Variable-input deep operator networks. 5 2022. URL http://arxiv.org/abs/2205.11404.

M. Raissi, P. Perdikaris, and G. 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, 2 2019. ISSN 10902716. doi: 10.1016/j.jcp.2018.10.045.

Jacob H Seidman, Georgios Kissas, Paris Perdikaris, and George J Pappas. Nomad: Nonlinear manifold decoders for operator learning. *Advances in Neural Information Processing Systems*, 2022. URL https://github.com/PredictiveIntelligenceLab/NOMAD.

Vincent Sitzmann, Eric R. Chan, Richard Tucker, Noah Snavely, and Gordon Wetzstein. Metasdf: Meta-learning signed distance functions. *Advances in Neural Information Processing Systems*, 2020-December, 2020a. ISSN 10495258.

Vincent Sitzmann, Julien N.P. Martel, Alexander W. Bergman, David B. Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. *Advances in Neural Information Processing Systems*, 2020-December, 2020b. ISSN 10495258.

Matthew Tancik, Pratul P. Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan T. Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. *Advances in Neural Information Processing Systems*, 2020-December, 2020. ISSN 10495258.

Matthew Tancik, Ben Mildenhall, Terrance Wang, Divi Schmidt, Pratul P. Srinivasan, Jonathan T. Barron, and Ren Ng. Learned initializations for optimizing coordinate-based neural representations. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2021. ISSN 10636919. doi: 10.1109/CVPR46437.2021.00287.

Yuan Yin, Matthieu Kirchmeyer, Jean-Yves Franceschi, Alain Rakotomamonjy, and Patrick Gallinari. Continuous pde dynamics forecasting with implicit neural representations. *International Conference on Learning Representations*, 9 2022. URL http://arxiv.org/abs/2209.14855.

Luisa Zintgraf, Kyriacos Shiarlis, Vitaly Kurin, Katja Hofmann, and Shimon Whiteson. Fast context adaptation via meta-learning. *36th International Conference on Machine Learning, ICML 2019*, 2019-June, 2019.

A APPENDIX

A.1 RELATED WORK

Operator Learning We present two popular methods for Operator Learning, an emerging research direction in Deep Learning for Physics that aims at learning mappings between infinite-dimensional functions. Neural Operator is a class of models that process inputs by iterative kernel integration over the spatial domain. This integration step, which is more expensive than a convolutional-layer and resembles an attention scheme, needs to be approximated to reduce the time complexity in grid size. FNO (Li et al., 2021) uses the FFT to approximate this integral, since a convolution in the observation domain is a multiplication in the Fourier domain. A second model family, DeepONet (Lu et al., 2021) is an Operator Network that formulates the operator directly as an affine combination of spatial functions with several extensions (Seidman et al. (2022), Prasthofer et al. (2022)).

Spatial INRs Spatial INRs are a class of coordinate-based Neural Networks that model data as the realization of an implicit function of a spatial location $x \in \Omega \mapsto f_\theta(x)$. They make use of positional embedding and sinusoidal activation to better capture high-frequency details (Tancik et al. (2020), Sitzmann et al. (2020b), Fathony et al. (2021), Lindell et al. (2022)). Vanilla INR architectures can be evaluated on any domain location without the need for data interpolation, but are compatible with only one data sample $u|_\mathcal{X}$. Prior works based on meta-learning (Tancik et al. (2021), Sitzmann et al. (2020a)), auto-encoders (Chen & Zhang (2019) Mescheder et al. (2019)), or modulation (Park et al. (2019), Dupont et al. (2022)) have tackled this limitation so that an INR with parameters θ can decode different functions $u_i|_\mathcal{X}$ with the help of per-sample parameter z_i. Yin et al. (2022) and Chen et al. (2022) used spatial INRs to model physical dynamics.

A.2 ADDITIONAL RESULTS ON INITIAL VALUE PROBLEM

We provide in this section additional results for the initial value problem task. First we complete the results on *Navier-Stokes*, and then present the performance of CORAL on the more challenging *Shallow-Water* dataset.

A.2.1 NAVIER-STOKES

We show in Table 3 the comparison of our method against Geo-FNO. Even in the sparse $\pi_{tr} = 5\%$ setting, CORAL is able to generalize to new initial conditions on denser grid. In comparison, the performance of Geo-FNO drops drastically when test samples are observed on a different grid. We display in Figure 3 the train grid used for the $\pi_{tr} = 5\%$ setting, and show the super-resolution capabilities of CORAL in Figure 4.

Table 3: **Irregular up-sampling** - Test results on *Navier-Stokes*. The model is trained on π_{tr} percent of a 128×128 grid. We write *n.a.* when the inference diverges.

$\mathcal{X}_{tr} \downarrow$	$\mathcal{X}_{te} \rightarrow$	128×128	256×256
$\pi_{tr} = 5\%$	Geo-FNO	*n.a.*	*n.a.*
	CORAL	**0.00566**	**0.0231**
$\pi_{tr} = 25\%$	Geo-FNO	1.161	*n.a.*
	CORAL	**0.00277**	**0.0227**

A.2.2 SHALLOW-WATER

• **Dataset**. We also assessed the performance of our model on the **3D-Spherical Shallow-Water equation** (*Shallow-Water*). This equation can be used as an approximation to a flow on the earth's surface. The data consists of the *Vorticity* w, and *Height* h of the fluid. We handle them separately and learn to map the vorticity at time $t_0 = 180$, ie. $w(\cdot, t_0 = 180)$ to the vorticity at time $T = 240$, i.e. $w(\cdot, T = 240)$; as well as the height at time $t_0 = 180$, i.e. $h(\cdot, t_0 = 180)$ to the height at time $T = 240$, i.e. $h(\cdot, T = 240)$. The train size is $n = 1000$ samples and test size is 200 samples.

For *Shallow-Water*, the *Vorticity* channel exhibits higher frequency patterns than *Height* and is thus more challenging to predict. Yet CORAL is able to capture the dynamics on both channels with

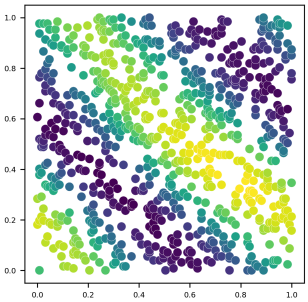


Figure 3: Visualization of the irregular grid for *Navier-Stokes*. Here we randomly subsampled $\pi_{tr} = 5\%$ of the 128×128 grid.

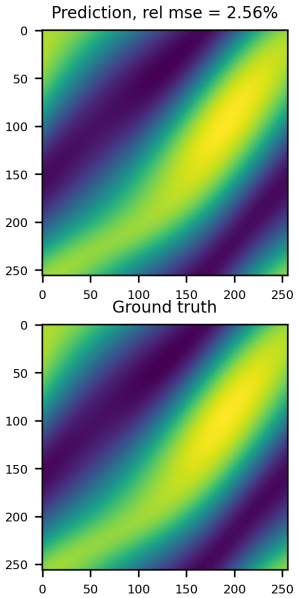
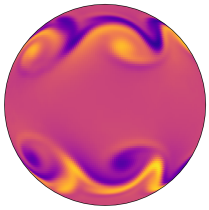


Figure 4: Visualization of super-resolution for *Navier-Stokes*. We train CORAL on a 64×64 grid and show the prediction for a test input of resolution 256×256 compared to the ground truth.

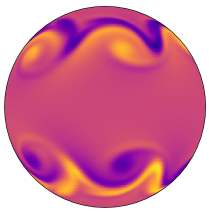
regular or irregular training grids. In Table 4, we can see it is on par with FNO when the test resolution stays the same, and obtains far superior results on the up-sampling settings.

Table 4: Test results on *Shallow-water*. (a) **Same-grid** $\mathcal{X}_{tr} = \mathcal{X}_{te}$ (b) **Regular up-sampling** with train resolution of 64×128 and test resolution of 128×256 (c) **Irregular up-sampling** with π_{tr} = 25 % and test resolution 128×256. For both (a) and (b) baseline is FNO as the grids are regular; and we use geo-FNO in (c). We write *n.a.* when the inference diverges.

	(a) 64×128		**(b)** 128×256		**(c)** $\pi_{tr} = 25\%$	
Model	*Height*	*Vorticity*	*Height*	*Vorticity*	*Height*	*Vorticity*
FNO \ Geo-FNO	0.00285	**0.00316**	0.156	0.651	*n.a.*	*n.a.*
CORAL	**0.00146**	0.00374	**0.00184**	**0.0118**	**0.00565**	**0.0399**



(a) Prediction



(b) Ground truth

Figure 5: Visualization of super-resolution on *Vorticity*. We train CORAL on a randomly sub-sampled grid with $\pi_{tr} = 25\%$ of a 128×256 grid and show the prediction for a test input of resolution 128×256 compared to the ground truth.

A.3 Dataset Details

2D-Navier-Stokes (*Navier-Stokes*) We consider the 2D Navier-Stokes equation for a viscous, incompressible fluid in vorticity form on the unit torus:

$$\frac{\partial w(x,t)}{\partial t} + u(x,t) \cdot \nabla w(x,t) = \nu \Delta w(x,t) + f(x), x \in]0,1[^2, t \in]0,T] \tag{4}$$

$$\nabla u(x,t) = 0, x \in]0,1]^2, t \in [0,T] \tag{5}$$

$$w(x,0) = w_0(x), x \in]0,1[^2 \tag{6}$$

where u is the velocity field, $w = \nabla \times u$ is the vorticity, w_0 is the initial vorticity, ν is the viscosity, and f is the forcing funtion. We want to learn the mapping $w(\cdot, t_0 = 10)$ to $w(\cdot, T = 20)$. We generate the data as in Li et al. (2021).

3D-Spherical Shallow-Water (*Shallow-Water*). We consider the global shallow-water equations:

$$\frac{du}{dt} = -f \cdot k \times u - g\nabla h + \nu \Delta u \tag{7}$$

$$\frac{dh}{dt} = -h\nabla \cdot u + \nu \Delta h \tag{8}$$

where $\frac{d}{dt}$ is the material derivative, k is the unit vector orthogonal to the spherical surface, u is the velocity field tangent to the surface of the sphere, which can be transformed into the vorticity $w = \nabla \times u$, h is the height of the sphere. We generate the data with the *Dedalus* software (Burns et al., 2020).

Euler's Equation (*Airfoil*) We consider the transonic flow over an airfoil, where the governing equation is Euler equation, as following,

$$\frac{\partial \rho_f}{\partial t} + \nabla \cdot (\rho_f \boldsymbol{u}) = 0, \frac{\partial \rho_f \boldsymbol{u}}{\partial t} + \nabla \cdot (\rho_f \boldsymbol{u} \otimes \boldsymbol{u} + p\mathbb{I}) = 0, \frac{\partial E}{\partial t} + \nabla \cdot ((E + p)\boldsymbol{u}) = 0, \quad (9)$$

where ρ_f is the fluid density, u is the velocity vector, p is the pressure, and E is the total energy. The viscous effect is ignored. We use the dataset from Li et al. (2022).

A.4 Implementation Details

Shift modulated Siren In practice, we observed no gain of performance in using both scale and shift modulations, and chose to stick with shift modulations only. The INR forward pass is thus written as:

$$\begin{cases} \boldsymbol{h}_0 = x \\ \boldsymbol{h}_{l+1} = \sin(\omega_0(\boldsymbol{W}_l \boldsymbol{h}_l + \boldsymbol{b}_i + \boldsymbol{B}_i z + \boldsymbol{d}_i)), \text{ for } 0 \leq l \leq L-1, \\ \tilde{y} = \boldsymbol{W}_L \boldsymbol{h}_L + \boldsymbol{b}_L \end{cases} \quad (10)$$

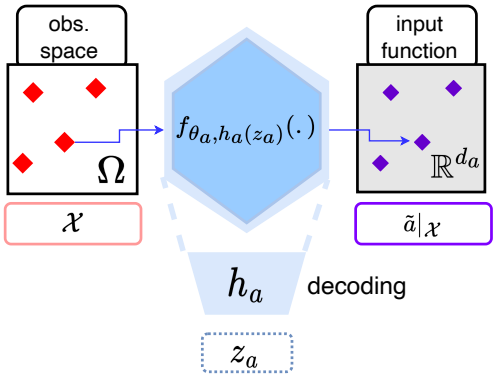


Figure 6: Architecture of the input decoder ξ_a. The hypernetwork h_a maps the code z_a to the parameter space of an INR $f_{\theta_a} \in \mathcal{A}$. We can query this INR on any coordinate $x \in \Omega$.

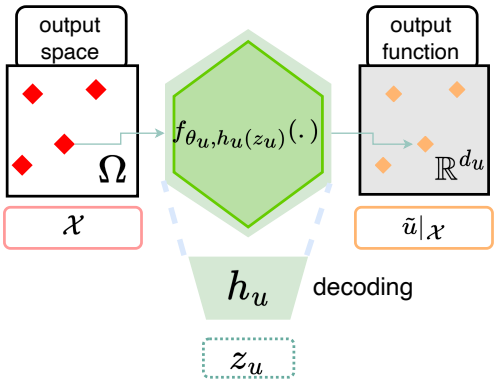


Figure 7: Architecture of the output decoder ξ_u. The hypernetwork h_u maps the code z_u to the parameter space of an INR $f_{\theta_u} \in \mathcal{U}$. We can query this INR on any coordinate $x \in \Omega$.

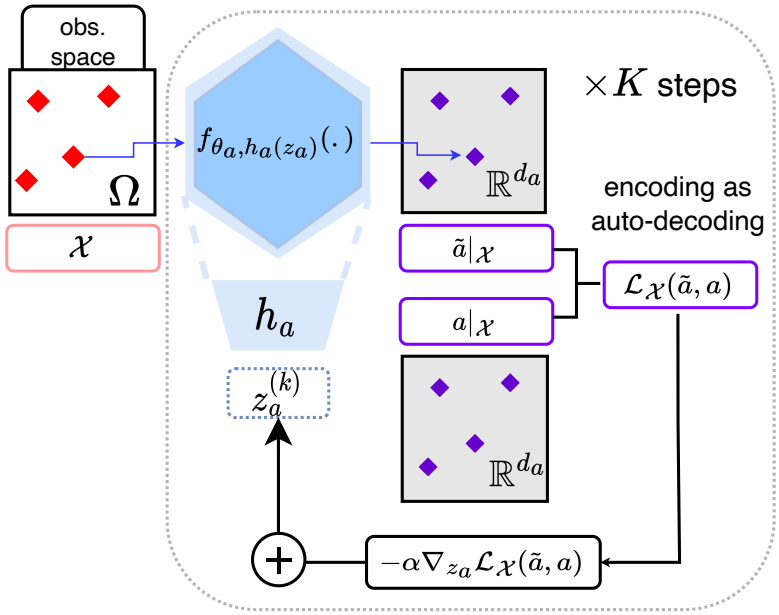


Figure 8: Starting from a code $z_a^{(0)} = 0$, the input encoder performs K inner steps of gradient descent over z_a to minimize the reconstruction loss $\mathcal{L}_{\mathcal{X}}(\tilde{a}, a)$ and outputs the resulting code $z_a^{(K)}$ of this optimization process. During training, we accumulate the gradients of this encoding phase and back-propagate through the K inner-steps to update the parameters θ_a and w_a. At inference, we encode new inputs with the same number of steps K and the same learning rate α, unless stated otherwise. The output encoder works in the same way during training, and is not used at inference.

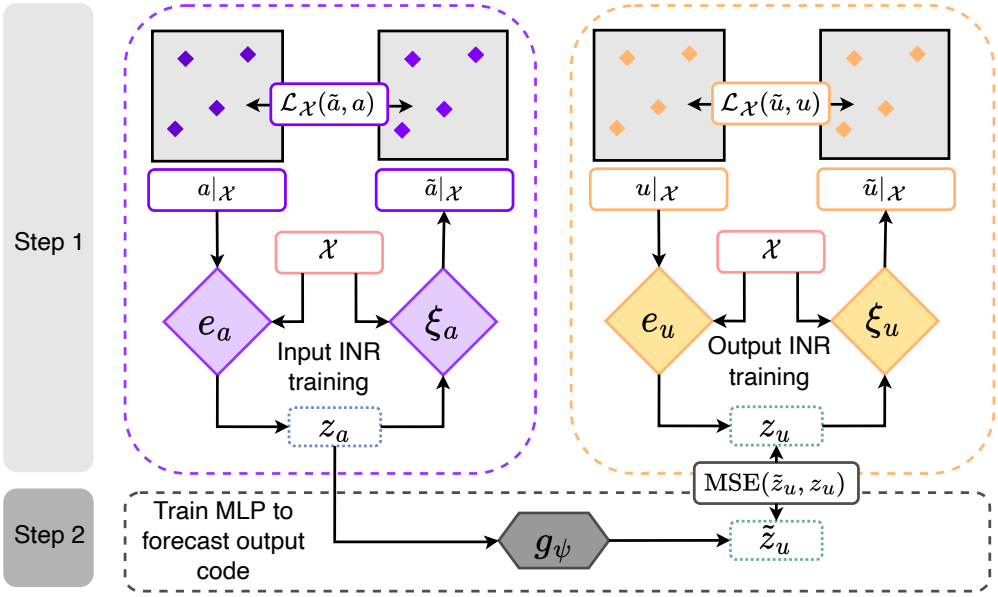


Figure 9: Proposed training for CORAL. For each epoch we perform, (1) a learning step to minimize the reconstruction loss over the inputs a_i and outputs u_i; (2) a learning step to minimize the MSE between the inferred codes $g_\psi(z_{a_i})$ and z_{u_i}.

Algorithm 1: CORAL Training

while not done **do**

  Sample batch $\mathcal{B}$ of data $(a_i, u_i)_{i\in\mathcal{B}}$;

  Set codes to zero $z_{a_i} \leftarrow 0, z_{u_i} \leftarrow 0, \forall i \in \mathcal{B}$;

  for $i \in \mathcal{B}$ and step $\in \{1, ..., K_a\}$ **do**

    $z_{a_i} \leftarrow z_{a_i} - \alpha_a \nabla_{z_{a_i}} \mathcal{L}_{\mathcal{X}_{tr}}(f_{\theta_a, h_a(z_{a_i})}, a_i)$; // input encoding inner step

  end

  for $i \in \mathcal{B}$ and step $\in \{1, ..., K_u\}$ **do**

    $z_{u_i} \leftarrow z_{u_i} - \alpha_u \nabla_{z_{u_i}} \mathcal{L}_{\mathcal{X}_{tr}}(f_{\theta_u, h_u(z_{u_i})}, u_i)$; // output encoding inner step

  end

  /* outer loop update for input */

  $\theta_a \leftarrow \theta_a - \eta \frac{1}{|\mathcal{B}|} \sum_{i\in\mathcal{B}} \nabla_{\theta_a} \mathcal{L}_{\mathcal{X}_{tr}}(f_{\theta_a, h_a(z_{a_i})}, a_i)$,

  $w_a \leftarrow w_a - \eta \frac{1}{|\mathcal{B}|} \sum_{i\in\mathcal{B}} \nabla_{w_a} \mathcal{L}_{\mathcal{X}_{tr}}(f_{\theta_a, h_a(z_{a_i})}, a_i)$

  /* outer loop update for output */

  $\theta_u \leftarrow \theta_u - \eta \frac{1}{|\mathcal{B}|} \sum_{i\in\mathcal{B}} \nabla_{\theta_u} \mathcal{L}_{\mathcal{X}_{tr}}(f_{\theta_u, h_u(z_{u_i})}, u_i)$,

  $w_u \leftarrow w_u - \eta \frac{1}{|\mathcal{B}|} \sum_{i\in\mathcal{B}} \nabla_{w_u} \mathcal{L}_{\mathcal{X}_{tr}}(f_{\theta_u, h_u(z_{u_i})}, u_i)$

  /* inference update */

  $\psi \leftarrow \psi - \eta_\psi \frac{1}{|\mathcal{B}|} \sum_{i\in\mathcal{B}} \nabla_\psi \mathcal{L}(g_\psi(z_{a_i}), z_{u_i})$;

end

Algorithm 2: CORAL Inference given a

Set code to zero $z_a \leftarrow 0$;

for step $\in \{1, ..., K_a\}$ **do**

  $z_a \leftarrow z_a - \alpha_a \nabla_{z_a} \mathcal{L}_{\mathcal{X}}(f_{\theta_a, h_a(z_a)}, a)$; // input encoding inner step

end

$\tilde{z}_u = g_\psi(z_a)$; // predict latent code

$\tilde{u} = f_{\theta_u, h_u(\tilde{z}_u)}$; // decode output function