

# Convolutional Neural Operators for robust and accurate learning of PDEs

Bogdan Raonić<sup>1,2</sup>, Roberto Molinaro<sup>1</sup>, Tim De Ryck<sup>1</sup>, Tobias Rohner<sup>1</sup>, Francesca Bartolucci<sup>3</sup>, Rima Alaifari<sup>1</sup>, Siddhartha Mishra<sup>1,2</sup>, and Emmanuel de Bézenac<sup>1</sup>

<sup>1</sup>Seminar for Applied Mathematics, ETH Zurich

<sup>2</sup>ETH AI Center

<sup>3</sup>Delft University of Technology

**ABSTRACT.** Although very successfully used in conventional machine learning, convolution based neural network architectures – believed to be inconsistent in function space – have been largely ignored in the context of learning solution operators of PDEs. Here, we present novel adaptations for convolutional neural networks to demonstrate that they are indeed able to process functions as inputs and outputs. The resulting architecture, termed as convolutional neural operators (CNOs), is designed specifically to preserve its underlying continuous nature, even when implemented in a discretized form on a computer. We prove a universality theorem to show that CNOs can approximate operators arising in PDEs to desired accuracy. CNOs are tested on a novel suite of benchmarks, encompassing a diverse set of PDEs with possibly multi-scale solutions and are observed to significantly outperform baselines, paving the way for an alternative framework for robust and accurate operator learning. Our code is publicly available at <https://github.com/bogdanraonic3/ConvolutionalNeuralOperator>.

## 1 Introduction.

Partial Differential Equations (PDEs) [13] are ubiquitous as mathematical models in the sciences and engineering. Solving a PDE amounts to (approximately) computing the so-called *solution operator* that maps function space inputs such as initial and boundary conditions, coefficients, source terms etc, to the PDE solution which also belongs to a suitable function space. Well-established numerical methods such as finite differences, finite elements, finite volumes and spectral methods (see [48]) have been very successfully used for many decades to approximate PDE solution operators. However, the prohibitive computational cost of these methods, particularly in high dimensions and for *many query* problems such as UQ, inverse problems, PDE-constrained control and optimization, necessitates the design of *fast, robust and accurate* surrogates. This provides the rationale for the use of *data-driven machine learning* methods for solving PDEs [21].

As *operators* are the objects of interest in solving PDEs, learning such operators from data, which is loosely termed as *operator learning*, has emerged as a dominant paradigm in recent

years for the applications of machine learning to PDEs. A very partial list of architectures for operator learning include operator networks [9], DeepONets [39] and its variants [41, 7], PCA-net [5], neural operators [25] such as graph neural operator [34], Multipole neural operator [35] and the very popular Fourier Neural Operator [33] and its variants [36, 45], VIDON [47], spectral neural operator [14], LOCA [23], NOMAD [52] and transformer based operator learning architectures [8].

Despite the considerable success of the recently proposed operator learning architectures, several pressing issues remain to be addressed. These include, but are by no means restricted to, limited expressivity for some of these algorithms [28] to aliasing errors for others [14] to the very fundamental issue of possible *lack of consistency in function spaces* for many of them. As argued in a recent paper [2], a structure-preserving operator learning algorithm or *representation equivalent neural operator* has to respect some form of *continuous-discrete equivalence* (CDE) in order to learn the underlying operator, rather than just a discrete representation of it. Failure to respect such a CDE can lead to the so-called *aliasing errors* [2] and affect model performance at multiple discrete resolutions.

Despite many attempts, see [1, 16] and references therein, the absence of a suitable CDE, resulting in aliasing errors, has also plagued the naive use of convolutional neural networks (CNNs) in the context of operator learning, see [65, 33, 2] on how using CNNs for operator learning leads to results that heavily rely on the underlying grid resolution. This very limited use of Convolution (in physical space) based architectures for operator learning stands in complete contrast to the fact that CNNs [30] and their variants are widely used architectures for image classification and generation and in other contexts in machine learning [29, 37, 63]. Moreover, CNNs can be thought of as natural generalizations of the foundational finite difference methods for discretizing PDEs [17, 38]. Given their innate locality, computational and data efficiency, ability to process multi-scale inputs and outputs and the availability of a wide variety of successful CNN architectures in other fields, it could be very advantageous to bring CNN-based algorithms back into the reckoning for operator learning. This is precisely the central point of the current paper where we make the following contributions,

- We propose novel modifications to CNNs in order to enforce structure-preserving continuous-discrete equivalence and enable the genuine, alias-free, learning of operators. The resulting architecture, termed as *Convolutional Neural Operator* (CNO), is instantiated as a novel *operator* adaptation of the widely used U-Net architecture.
- In addition to showing that CNO is a *representation equivalent neural operator* in the sense of [2], we also prove a universality result to rigorously demonstrate that CNOs can approximate the operators, corresponding to a large class of PDEs, to desired accuracy.
- We test CNO on a *novel* set of benchmarks, that we term as *Representative PDE Benchmarks* (RPB), that span across a variety of PDEs ranging from linear elliptic and hyperbolic to nonlinear parabolic and hyperbolic PDEs, with possibly *multiscale solutions*. We find that CNO is either on-par or outperforms the tested baselines on all the benchmarks, both when testing in-distribution as well as in *out-of-distribution* testing.

Thus, we present a new CNN-based operator learning model, with desirable theoretical properties and excellent empirical performance, with the potential to be widely used for learning PDEs.

## 2 Convolutional Neural Operators.

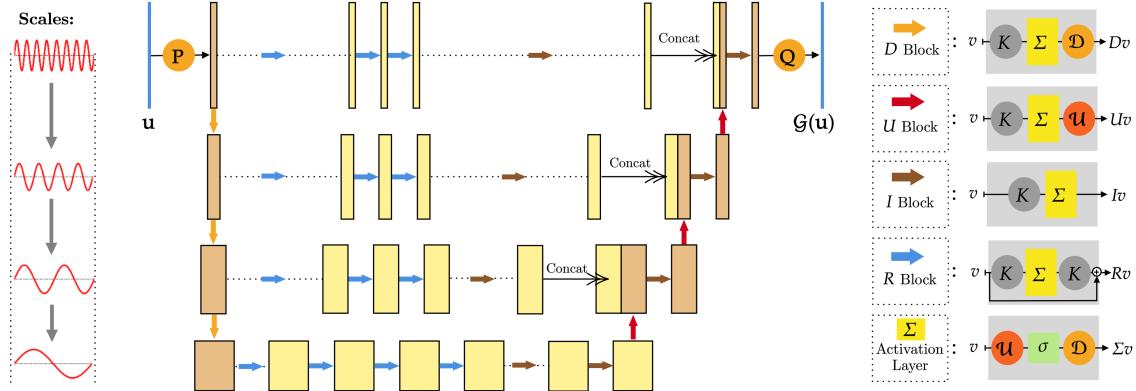


Figure 1: Schematic representation of CNO (2.3) as a modified U-Net with a sequence of layers (each identified with the relevant operators on the right, see Section 2) mapping between bandlimited functions. Rectangles represent multi-channel signals. Larger the height, larger is the resolution. Wider the rectangles, more channels are present.

**Setting.** For simplicity of the exposition, we will focus here on the two-dimensional case by specifying the underlying domain as  $D = \mathbb{T}^2$ , being the 2-d torus. Let  $\mathcal{X} = H^r(D, \mathbb{R}^{d_x}) \subset \mathcal{Z}$  and  $\mathcal{Y} = H^s(D, \mathbb{R}^{d_y})$  be the underlying function spaces, where  $H^{r,s}(D, \cdot)$  are Sobolev spaces of order  $r$  and  $s$ . Without loss of generality, we set  $r = s$  hereafter. Our aim would be to approximate *continuous operators*  $\mathcal{G}^\dagger : \mathcal{X} \rightarrow \mathcal{Y}$  from data pairs  $(u_i, \mathcal{G}^\dagger(u_i))_{i=1}^M \in \mathcal{X} \times \mathcal{Y}$ . We further assume that there exists a *modulus of continuity* for the operator i.e.,

$$\|\mathcal{G}^\dagger(u) - \mathcal{G}^\dagger(v)\|_{\mathcal{Y}} \leq \omega(\|u - v\|_{\mathcal{Z}}), \quad \forall u, v \in \mathcal{X}, \quad (2.1)$$

with  $\omega : \mathbb{R}_+ \rightarrow \mathbb{R}_+$  being a monotonically increasing function with  $\lim_{y \rightarrow 0} \omega(y) = 0$ . The underlying operator  $\mathcal{G}^\dagger$  can correspond to solution operators for PDEs (see Section 3 for the exact setting) but is more general than that and encompasses examples such as those arising in inverse problems, for instance in imaging [4].

**Bandlimited Approximation.** As argued in a recent paper [2], Sobolev spaces such as  $H^r$  are, in a sense, *too large* to allow for any form of *continuous-discrete equivalence* (CDE), i.e., equivalence between the underlying operator and its discrete representations, which is necessary for robust operator learning. Consequently, one has to consider smaller subspaces of  $H^r$  which allow for such CDEs. In this respect, we choose the space of *bandlimited functions* [59] defined

by,

$$\mathcal{B}_w(D) = \{f \in L^2(D) : \text{supp} \hat{f} \subseteq [-w, w]^2\}, \quad (2.2)$$

for some  $w > 0$  and with  $\hat{f}$  denoting the Fourier transform of  $f$ . It is straightforward to show using (2.1) (see **SM A.1**) that for any  $\varepsilon > 0$ , there exists a  $w$ , large enough depending on  $r$ , and a continuous operator  $\mathcal{G}^* : \mathcal{B}_w(D) \rightarrow \mathcal{B}_w(D)$ , such that  $\|\mathcal{G}^\dagger - \mathcal{G}^*\| < \varepsilon$ , with  $\|\cdot\|$  denoting the corresponding operator norm. In other words, the underlying operator  $\mathcal{G}^\dagger$  can be approximated to arbitrary accuracy by the operator  $\mathcal{G}^*$  that maps between band-limited spaces. Consequently, as shown in **SM A.2**, one can readily define discrete versions of  $\mathcal{G}^*$  using the underlying *sinc* basis for bandlimited functions and establish a continuous-discrete equivalence for it.

**Definition of CNO.** Given the above context, our goal will be to approximate the operator  $\mathcal{G}^*$  in a *structure-preserving manner* i.e., as the underlying operator maps between spaces of bandlimited functions, we will construct our operator approximation architecture to also map bandlimited functions to bandlimited functions, thus respecting the continuous-discrete equivalence. To this end, we denote the operator  $\mathcal{G} : \mathcal{B}_w(D) \rightarrow \mathcal{B}_w(D)$  as a *convolutional neural operator* (CNO) which we define as a compositional mapping between functions as

$$\mathcal{G} : u \mapsto P(u) = v_0 \mapsto v_1 \mapsto \dots v_L \mapsto Q(v_L) = \bar{u}, \quad (2.3)$$

where

$$v_{l+1} = \mathcal{P}_l \circ \Sigma_l \circ \mathcal{K}_l(v_l), \quad 1 \leq l \leq L-1. \quad (2.4)$$

From (2.3), we see that first, the input function  $u \in \mathcal{B}_w(D)$  is lifted to the latent space of bandlimited functions through a *lifting layer*:

$$P : \left\{ u \in \mathcal{B}_w(D, \mathbb{R}^{d_x}) \right\} \rightarrow \left\{ v_0 \in \mathcal{B}_w(D, \mathbb{R}^{d_0}) \right\}.$$

Here,  $d_0 > d_x$  is the number of channels in the lifted, latent space. The lifting operation is performed by a convolution operator which will be defined below.

Then, the lifted function is processed through the composition of a series of mappings between functions (layers), with each layer consisting of three elementary mappings, i.e.,  $\mathcal{P}_l$  is either the *upsampling* or *downsampling* operator,  $\mathcal{K}_l$  is the convolution operator and  $\Sigma_l$  is the activation operator. These elementary operators are defined below and are inspired by the modifications of CNNs for image generation in [22]. Finally, the last output function in the iterative procedure  $v_L$  is projected to the output space with a *projection operator*  $Q$ , defined as

$$Q : \left\{ v_L \in \mathcal{B}_w(D, \mathbb{R}^{d_L}) \right\} \rightarrow \left\{ \bar{u} \in \mathcal{B}_w(D, \mathbb{R}^{d_y}) \right\}.$$

The projection operation is also performed by a convolution operator defined below.

**Convolution Operator.** For simplicity of exposition, we will present the *single-channel* version of the convolution operator  $\mathcal{K}_l$  here. See **SM A.3** for the *multi-channel* version for this

and other operators considered below. Convolution operations are performed with discrete kernels

$$K_w = \sum_{i,j=1}^k k_{ij} \cdot \delta_{z_{ij}}$$

defined on the  $s \times s$  uniform grid on  $D$  with grid size  $\leq 1/2w$ , in-order to satisfy the requirements of the Whittaker-Shannon-Kotelnikov sampling theorem [58], and  $z_{ij}$  being the resulting grid points,  $k \in \mathbb{N}$  being the kernel size and  $\delta_x$  denoting the Dirac measure at point  $x \in D$ . The convolution operator for a *single-channel*  $\mathcal{K}_w : \mathcal{B}_w(D) \rightarrow \mathcal{B}_w(D)$  is defined by

$$\mathcal{K}_w f(x) = (K_w \star f)(x) = \int_D K_w(x-y) f(y) dy = \sum_{i,j=1}^k k_{ij} f(x - z_{ij}), \quad \forall x \in D,$$

where the last identity arises from the fact that  $f \in \mathcal{B}_w$ . Thus, our convolution operator is directly parametrized in physical space, in contrast to the Fourier space parametrization of a convolution in the FNO architecture of [33]. Hence, our parametrization is of a *local* nature.

**Upsampling and Downsampling Operators.** For some  $\bar{w} > w$ , we can *upsample* a function  $f \in \mathcal{B}_w$  to the *higher band*  $\mathcal{B}_{\bar{w}}$  by simply setting,

$$\mathcal{U}_{w,\bar{w}} : \mathcal{B}_w(D) \rightarrow \mathcal{B}_{\bar{w}}(D), \quad \mathcal{U}_{w,\bar{w}} f(x) = f(x), \quad \forall x \in D.$$

On the other hand, for some  $\underline{w} < w$ , we can *downsample* a function  $f \in \mathcal{B}_w$  to the *lower band*  $\mathcal{B}_{\underline{w}}$  by setting  $\mathcal{D}_{w,\underline{w}} : \mathcal{B}_w(D) \rightarrow \mathcal{B}_{\underline{w}}(D)$ , defined by

$$\mathcal{D}_{w,\underline{w}} f(x) = \left(\frac{\underline{w}}{w}\right)^2 (h_{\underline{w}} \star f)(x) = \left(\frac{\underline{w}}{w}\right)^2 \int_D h_{\underline{w}}(x-y) f(y) dy, \quad \forall x \in D,$$

where  $\star$  is the convolution operation on functions defined above and  $h_{\underline{w}}$  is the so-called *interpolation sinc filter*:

$$h_w(x_0, x_1) = \text{sinc}(2wx_0) \cdot \text{sinc}(2wx_1), \quad (x_0, x_1) \in \mathbb{R}^2. \quad (2.5)$$

**Activation Layer.** Naively, one can apply the activation function pointwise to any function. However, it is well-known that such an application will no longer respect the band-limits of the underlying function space and generate *aliasing errors* [22, 14, 2]. In particular, nonlinear activations can generate features at arbitrarily high frequencies. As our aim is to respect the underlying CDE, we will modulate the application of the activation function so that the resulting outputs fall within desired band limits. To this end, we first upsample the input function  $f \in \mathcal{B}_w$  to a higher bandlimit  $\bar{w} > w$ , then apply the activation and finally downsample the result back to the original bandlimit  $w$  (See Figure 1). Implicitly assuming that  $\bar{w}$  is large enough such that  $\sigma(\mathcal{B}_w) \subset \mathcal{B}_{\bar{w}}$ , we define the activation layer in (2.3) as,

$$\Sigma_{w,\bar{w}} : \mathcal{B}_w(D) \rightarrow \mathcal{B}_w(D), \quad \Sigma_{w,\bar{w}} f(x) = \mathcal{D}_{\bar{w},w}(\sigma \circ \mathcal{U}_{w,\bar{w}} f)(x), \quad \forall x \in D. \quad (2.6)$$

**Instantiation through an Operator U-Net architecture.** The above ingredients are assembled together in the form of an Operator U-Net architecture that has bandlimited functions as inputs and outputs. In addition to the blocks that have been defined above, we also need additional ingredients, namely incorporate *skip connections* through *ResNet* blocks of the form,  $\mathcal{R}_{w,\bar{w}} : \mathcal{B}_w(D, \mathbb{R}^d) \rightarrow \mathcal{B}_w(D, \mathbb{R}^d)$  such that

$$\mathcal{R}_{w,\bar{w}}(v) = v + \mathcal{K}_w \circ \Sigma_{w,\bar{w}} \circ \mathcal{K}_w(v), \quad \forall v \in \mathcal{B}_w(D, \mathbb{R}^d). \quad (2.7)$$

We also need the so-called *Invariant blocks* of the form,  $\mathcal{I}_{w,\bar{w}} : \mathcal{B}_w(D, \mathbb{R}^d) \rightarrow \mathcal{B}_w(D, \mathbb{R}^d)$  such that

$$\mathcal{I}_{w,\bar{w}}(v) = \Sigma_{w,\bar{w}} \circ \mathcal{K}_w(v), \quad \forall v \in \mathcal{B}_w(D, \mathbb{R}^d). \quad (2.8)$$

Finally, all these ingredients are assembled together in a modified Operator U-Net architecture which is graphically depicted in Figure 1. As seen from this figure, the input function, say  $u \in \mathcal{B}_w(D, \mathbb{R}^{d_x})$  is first lifted and then processed through a series of layers. Four types of blocks are used i.e., downsampling (D) block corresponding to using the downsampling operator  $\mathcal{D}$  as the  $\mathcal{P}$  in (2.4), upsampling (U) block corresponding to using the upsampling operator  $\mathcal{U}$  as the  $\mathcal{P}$  in (2.4), ResNet (R) block corresponding to (2.7) and Invariant (I) block corresponding to (2.8). Each block takes a band-limited function as input and returns another band-limited function (with the same band) as the output. Finally, U-Net style patching operators, which concatenate outputs for different layers as additional channels are also used. As these operations act only in the channel width and leave the spatial resolution unchanged, they conform to the underlying bandlimits. Thus, CNO takes a function input and passes it through a set of encoders, where the input is downsampled in space but expanded in channel width and then processed through a set of decoders, where the channel width is reduced but the space resolution is increased. At the same time, encoder and decoder layers (at the same spatial resolution or band limit) are connected through additional ResNet blocks. Thus, this architectural choice allows for transferring high frequency content via the skip connections, before filtering them out with the *sinc* filter as we go deeper into the encoder. Hence, the high frequency content is not just recreated with the activation function, but also modified through the intermediate networks. Consequently, we build a genuinely *multiscale operator learning architecture*.

**Continuous-Discrete Equivalence for CNO.** We have defined CNO (2.3) as an operator that maps bandlimited functions to bandlimited functions. In practice, like any computational algorithm, CNO has to be implemented in a discrete manner, with *discretized versions* of each of the above-defined elementary operations being specified in SM A.4. Given how each of the elementary blocks (convolution, up- and downsampling, activation, ResNets etc) are constructed, we prove the following proposition (in SM A.5):

**Proposition 2.1.** *Convolutional Neural Operator  $\mathcal{G} : \mathcal{B}_w(D, \mathbb{R}^{d_x}) \rightarrow \mathcal{B}_w(D, \mathbb{R}^{d_y})$  (2.3) is a Representation equivalent neural operator or ReNO, in the sense of [2], Definition 3.4.*

Further details about the notion of ReNOs is provided in SM A.4 and we refer the reader to [2], where this concept is presented in great detail and the representation equivalence of CNO is

discussed. In particular, following [2], representation equivalence implies that CNO satisfies a form of *resolution invariance*, allowing it to be evaluated on multiple grid resolutions without aliasing errors.

### 3 Universal Approximation by CNOs.

We want to prove that a large class of operators, stemming from PDEs, can be approximated to desired accuracy by CNOs. To this end, we consider the following abstract PDE in the domain  $D = \mathbb{T}^2$ ,

$$\mathcal{L}(u) = 0, \quad \mathcal{B}(u) = 0, \quad (3.1)$$

with  $\mathcal{L}$  being a differential operator and  $\mathcal{B}$  a boundary operator. We assume that the differential operator  $\mathcal{L}$  only depends on the coordinate  $x$  through a *coefficient* function  $a \in H^r(D)$ . The corresponding *solution* operator is denoted by  $\mathcal{G}^\dagger : \mathcal{X}^* \subset H^r(D) \rightarrow H^r(D) : a \mapsto u$ , with  $u$  being the solution of the PDE (3.1). We assume that  $\mathcal{G}^\dagger$  is continuous. Moreover, we also assume the following modulus of continuity,

$$\left\| \mathcal{G}^\dagger(a) - \mathcal{G}^\dagger(a') \right\|_{L^p(\mathbb{T}^2)} \leq \omega(\|a - a'\|_{H^\sigma(\mathbb{T}^2)}), \quad (3.2)$$

for some  $p \in \{2, \infty\}$  and  $0 \leq \sigma \leq r - 1$ , and where  $\omega : [0, \infty) \rightarrow [0, \infty)$  is a monotonously increasing function with  $\lim_{y \rightarrow 0} \omega(y) = 0$ . (3.2) is automatically satisfied if  $\mathcal{X}^*$  is compact and  $\mathcal{G}^\dagger$  is continuous. Under these assumptions, we have the following *universality theorem* for CNOs (2.3),

**Theorem 3.1.** *Let  $\sigma \in \mathbb{N}_0$  and  $p \in \{2, \infty\}$  as in (3.2),  $r > \max\{\sigma, 2/p\}$  and  $B > 0$ . For any  $\varepsilon > 0$  and any operator  $\mathcal{G}^\dagger$ , as defined above, there exists a CNO  $\mathcal{G}$  such that for every  $a \in \mathcal{X}^*$  with  $\|a\|_{H^r(D)} \leq B$  it holds,*

$$\|\mathcal{G}^\dagger(a) - \mathcal{G}(a)\|_{L^p(D)} < \varepsilon. \quad (3.3)$$

In fact, we will prove a more general version of this theorem in **SM B**, where we also include additional source terms in the PDE (3.1).

### 4 Experiments.

**Training Details and Baselines.** We provide a detailed description of the implementation of CNO and the training (and test) protocol for CNO as well as all the baselines in **SM C.1**. To ensure a *level playing field* among all the tested models for each benchmark, we follow an *ensemble training procedure* by specifying a range for the underlying hyperparameters *for each model* and randomly selecting a subset of the hyperparameter space. For each such hyperparameter configuration, the corresponding models are trained on the benchmark and the configuration with smallest validation error is selected and the resulting test errors are reported, allowing us to identify and compare the *best performing* version of each model for every benchmark. We

compare CNO with the following baselines: two very popular operator learning architectures, namely DeepONet (DON) [39] and FNO [33], a transformer based operator-learning architecture, i.e., Galerkin Transformer (GT) [8], feedforward neural network with residual connections (FFNN) [18] and the very widely-used ResNet [18] and U-Net [50] architectures. <sup>1</sup>

Table 1: Relative median  $L^1$  test errors, for both in- and out-of-distribution testing, for different benchmarks and models.

	In/Out	FFNN	GT	UNet	ResNet	DON	FNO	CNO
<b>Poisson Equation</b>	In	5.74%	2.77%	0.71%	0.43%	12.92%	4.98%	<b>0.21%</b>
	Out	5.35%	2.84%	1.27%	1.10%	9.15%	7.05%	<b>0.27%</b>
<b>Wave Equation</b>	In	2.51%	1.44%	1.51%	0.79%	2.26%	1.02%	<b>0.63%</b>
	Out	3.01%	1.79%	2.03%	1.36%	2.83%	1.77%	<b>1.17%</b>
<b>Smooth Transport</b>	In	7.09%	0.98%	0.49%	0.39%	1.14%	0.28%	<b>0.24%</b>
	Out	650.6%	875.4%	1.28%	0.96%	157.2%	3.90%	<b>0.46%</b>
<b>Discontinuous Transport</b>	In	13.0%	1.55%	1.31%	<b>1.01%</b>	5.78%	1.15%	<b>1.01%</b>
	Out	257.3%	22691.1%	1.35%	1.16%	117.1%	2.89%	<b>1.09%</b>
<b>Allen-Cahn Equation</b>	In	18.27%	0.77%	0.82%	1.40%	13.63%	<b>0.28%</b>	0.54%
	Out	46.93%	2.90%	2.18%	3.74%	19.86%	<b>1.10%</b>	2.23%
<b>Navier-Stokes Equations</b>	In	8.05%	4.14%	3.54%	3.69%	11.64%	3.57%	<b>2.76%</b>
	Out	16.12%	11.09%	10.93%	9.68%	15.05%	9.58%	<b>7.04%</b>
<b>Darcy Flow</b>	In	2.14%	0.86%	0.54%	0.42%	1.13%	0.80%	<b>0.38%</b>
	Out	2.23%	1.17%	0.64%	0.60%	1.61%	1.11%	<b>0.50%</b>
<b>Compressible Euler</b>	In	0.78%	2.09%	0.38%	1.70%	1.93%	0.44%	<b>0.35%</b>
	Out	1.34%	2.94%	0.76%	2.06%	2.88%	0.69%	<b>0.59%</b>

**Representative PDE Benchmarks (RPB).** Given the lack of consensus on a *standard* set of benchmarks for machine learning of PDEs, we propose a *new suite of benchmarks* here. Our aims in this regard are to ensure i) sufficient diversity among the types of PDE considered, ii) access to training and test data is readily available for rapid prototyping and reproducibility and iii) *intrinsic computational complexity* of problem to make sure that it is worthwhile to design fast surrogates to classical PDE solvers for a particular problem. In other words, we will only consider PDEs where classical PDE solvers can *only* resolve the underlying operator on fine enough grids. To meet these requirements, we will not consider PDEs in one space dimension as traditional numerical methods are already quite fast for them. On the other hand, it is hard to obtain and store data for problems in three dimensions, due to computational expense of traditional methods. The *sweet spot* is achieved by considering PDEs in two space dimensions. We further restrict to Cartesian domains here as all models can be readily evaluated in this

<sup>1</sup>The code can be found at <https://github.com/bogdanraonic3/ConvolutionalNeuralOperator>

setting. In addition to including a diverse set of PDEs, we only consider problems with sufficiently *many spatial and temporal scales*. Otherwise, traditional numerical solvers can approximate the underlying PDE on very coarse grids and it is not worthwhile to design surrogates (see **SM C.3.8** for a discussion in this context on a widely used Navier-Stokes benchmark). With these considerations in mind, we present the following subset of *Representative PDE Benchmarks* or **RPB**,

**Poisson Equation.** This prototypical *linear elliptic PDE* is given by,

$$-\Delta u = f, \text{ in } D, \quad u|_{\partial D} = 0. \quad (4.1)$$

The solution operator  $\mathcal{G}^\dagger : f \mapsto u$ , maps the source term  $f$  to the solution  $u$ . With source term,

$$f(x, y) = \frac{\pi}{K^2} \sum_{i,j=1}^K a_{ij} \cdot (i^2 + j^2)^{-r} \sin(\pi i x) \sin(\pi j y), \quad \forall (x, y) \in D, \quad (4.2)$$

with  $r = -0.5$ , the corresponding exact solution can be analytically computed (see **SM C.3.1**) and represents  $K$ - spatial scales.

For training the models, we fix  $K = 16$  in (4.2) and choose  $a_{ij}$  to be i.i.d. uniformly distributed from  $[-1, 1]$  (See **SM D** for a representation of the inputs and outputs of  $\mathcal{G}^\dagger$ ). This *multiscale solution* needs fine enough grid size to be approximated accurately by finite element methods, fitting our complexity criterion for benchmarks. In addition to *in-distribution* testing, we also consider an *out-of-distribution* testing task by setting  $K = 20$  in (4.2). This will enable us to evaluate the ability of the models to *generalize* to inputs (and outputs) with frequencies higher than those encountered during training.

**Wave Equation.** This prototypical *linear hyperbolic PDE* is given by

$$u_{tt} - c^2 \Delta u = 0, \text{ in } D \times (0, T), \quad u_0(x, y) = f(x, y), \quad (4.3)$$

with a constant propagation speed  $c = 0.1$ . The underlying operator  $\mathcal{G}^\dagger : f \mapsto u(., T)$  maps the initial condition  $f$  into the solution at the final time. If we consider initial conditions to be given by (4.2) with  $r = 1$ , then one can explicitly compute the exact solution (see **SM C.3.2**) to represent a *multiscale standing wave* with periodic pulsations (depending on  $K$ ) in time. The training and *in-distribution* test samples are generated by setting  $T = 5$ ,  $K = 24$  and

$a_{ij}$  to be i.i.d. uniformly distributed from  $[-1, 1]$  (See **SM D** for input and output samples). For *out-of-distribution* testing, we change the exponent of decay of the modes in (4.2) to  $r = 0.85$  and  $K = 32$ , in order to test the ability of the models to generalize to learn the effect of higher frequencies, than those present in the training data.

**Transport Equation.** The transport of scalar quantities of interest is modeled by PDE,

$$u_t + v \cdot \nabla u = 0, \quad u(t = 0) = f, \quad (4.4)$$

with a given velocity field and initial data  $f$ . The underlying operator  $\mathcal{G}^\dagger : f \mapsto u(., T = 1)$  maps the initial condition  $f$  into the solution at the final time. We set a constant velocity field  $v = (v_x, v_y) = (0.2, 0.2)$  leading to solution  $u(x, y, t) = f(x - v_x t, y - v_y t)$ . Two different types of training data are considered, i.e., *smooth* initial data which takes the form of a radially symmetric Gaussian, with centers randomly and uniformly drawn from  $(0.2, 0.4)^2$  and corresponding variance drawn uniformly from  $(0.003, 0.009)$  and a *discontinuous* initial data in the form of the indicator function of radial disk with centers, uniformly drawn from  $(0.2, 0.4)^2$  and radii uniformly drawn from  $(0.1, 0.2)$  (See SM C.3.3 for details and SM D for illustrations). For *out-of-distribution* testing in the smooth case, the centers of the Gaussian inputs are sampled uniformly from  $(0.4, 0.6)^2$  and in the discontinuous case, the centers of the disk are drawn uniformly from  $(0.4, 0.6)^2$ , while keeping the variance and the radii, respectively, the same as that of *in-distribution testing*. This *out-of-distribution* task tests the model's ability to cope with input translation-equivariance.

**Allen-Cahn Equation.** It is a prototype for *nonlinear parabolic PDEs*,

$$u_t = \Delta u - \varepsilon^2 u(u^2 - 1), \quad (4.5)$$

with a reaction rate of  $\varepsilon = 220$  and underlying operator  $\mathcal{G}^\dagger : f \mapsto u(., T)$ , mapping initial conditions  $f$  to the solution  $u$  at a final time  $T = 0.0002$ . The initial conditions for training and *in-distribution* testing are of the form (4.2), with  $r = 1$  and  $K = 24$  and coefficients  $a_{ij}$  drawn uniformly from  $[-1, 1]$ . For *out-of-distribution* testing, we set  $K = 16$  and randomly select the initial decay  $r$ , uniformly from the range  $[0.85, 1.15]$  of the modes in (4.2), which allows us to test the ability of the model to generalize to different dynamics of the system. Both training and test data are generated by using a finite difference scheme [64] on a grid at  $64^2$  resolution (see SM D for illustrations).

**Navier-Stokes Eqns.** These PDEs model the motion of incompressible fluids by,

$$u_t + (u \cdot \nabla)u + \nabla p = \nu \Delta u, \quad \operatorname{div} u = 0, \quad (4.6)$$

in the torus  $D = \mathbb{T}^2$  with periodic boundary conditions and viscosity  $\nu = 4 \times 10^{-4}$ , only applied to high-enough Fourier modes (those with amplitude  $\geq 12$ ) to model fluid flow at *very high Reynolds-number*. The solution operator  $\mathcal{G}^\dagger : f \mapsto u(., T)$ , maps the initial conditions  $f : D \rightarrow \mathbb{R}^2$  to the solution at final time  $T = 1$ . We consider initial conditions representing the well-known *thin shear layer* problem [3, 27] (See SM C.3.5 for details), where the shear layer evolves via vortex shedding to a complex distribution of vortices (see SM D for samples). The training and *in-distribution testing* samples are generated, with a spectral viscosity method [27], from an initial sinusoidal perturbation of the shear layer [27], with layer thickness  $\rho = 0.1$  and 10 perturbation modes, each sampled uniformly from  $[-1, 1]$ . For *out-of-distribution* testing, the layer thickness is reduced to  $\rho = 0.09$  and the layers are shifted up in the domain to test the ability of the models to *generalize* to a flow regime with an increased number and different locations of the shed vortices.

**Darcy flow.** The steady-state Darcy flow is described by the second order linear elliptic PDE,

$$-\nabla \cdot (a \nabla u) = f, \text{ in } D, \quad u|_{\partial D} = 0, \quad (4.7)$$

where  $a$  is the diffusion coefficient and  $f$  is the forcing term. We set the forcing term to  $f = 1$ . The solution operator is  $\mathcal{G}^\dagger : a \mapsto u$ , where the input is the diffusion coefficient  $a \sim \psi \# \mu$ , with  $\mu$  being a Gaussian Process with zero mean and squared exponential kernel

$$k(x, y) = \sigma^2 \exp\left(\frac{|x - y|^2}{l^2}\right), \quad \sigma^2 = 0.1. \quad (4.8)$$

We chose the length scale  $l = 0.1$  for the in-distribution testing and  $l = 0.05$  for the out-of-distribution testing. The mapping  $\psi : \mathbb{R} \rightarrow \mathbb{R}$  takes the value 12 on the positive part of the real line and 3 on the negative part. The push-forward measure is defined pointwise. The experimental setup is the same as the one presented in [33].

**Flow past airfoils.** We model this flow by the compressible Euler equations,

$$u_t + \operatorname{div} F(u) = 0, \quad u = [\rho, \rho v, E]^\perp, \quad F = [\rho v, \rho v \otimes v + p\mathbf{I}, (E + p)v]^\perp, \quad (4.9)$$

with density  $\rho$ , velocity  $v$ , pressure  $p$  and total Energy  $E$  related by an ideal gas equation of state. The airfoils we consider are described by perturbing the shape of a well-known RAE2822 airfoil [40] by Hicks-Henne Bump functions [42] (see SM C.3.7). Freestream boundary conditions are imposed and the solution operator maps the shape function onto the steady state density distribution (see SM D for samples) and training data are obtained with a compressible flow solver (NUWTUN) with shapes corresponding to 20 bump functions, with coefficients sampled uniformly from  $[0, 1]$ . *Out-of-distribution* testing is performed with 30 bump functions.

**Results.** The test errors, for both *in-distribution* and *out-of-distribution* testing for all the models on the **RPB** benchmarks are shown in Table 1. Starting with the *in-distribution* results, we see that among the baselines, FNO clearly outperforms both FFNN and DeepONet on all the **RPB** benchmarks as well as the Galerkin Transformer on all except the Poisson test case. On the other hand, the convolution-based U-Net and ResNet models are quite competitive vis-a-vis FNO, with comparable performances on most benchmarks, while outperforming FNO by a factor of 7 – 9 for the Poisson test case. This already indicates that convolution-based architectures can perform very well. Moreover, we observe from Table 1 that CNO is the best performing architecture on every task except Allen-Cahn. It readily outperforms FNO, for instance by almost a factor of 20 on the Poisson test case but more moderately but significantly on other tasks. It also outperforms U-Net and ResNet on all tasks considered here, emerging as the best-performing model over these benchmarks.

**Out-of-Distribution Testing.** This trend is further reinforced when we consider *out-of-distribution* testing. CNO generalizes well to unseen data in a *zero-shot* mode, with test errors

increasing by approximately a factor of 2, at most (with Allen-Cahn being an outlier) and still outperforms the baselines significantly in all cases other than Allen-Cahn, where FNO generalizes the best. FNO shows decent generalization for most problems but generalizes poorly on the transport problems. This can be attributed to its lack of translation equivariance, in contrast to U-Net, ResNet and CNO. Finally, the lack of translation invariance severely limits the out-of-distribution generalization performance of DeepONet, FFNN and Galerkin Transformer models on transport problems.

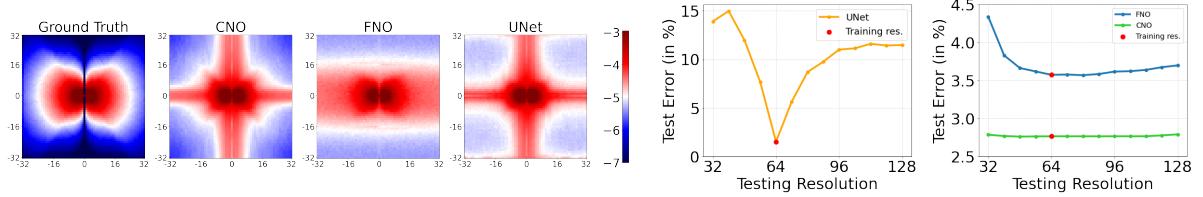


Figure 2: Thin Shear Layer Left: Averaged *logarithmic* amplitude spectra comparing Ground Truth, CNO, FNO and UNet. Right: Test error vs. Resolution for UNet, FNO and CNO.

**Resolution Invariance.** We select three of the best-performing models (U-Net, FNO and CNO) and highlight further differences between them for the Navier-Stokes test case (see also [SM C](#)). To this end, we start with Figure 2 (left), where we present the averaged (log) spectra for the ground truth (reference solution computed with the spectral viscosity method) and those computed with CNO, FNO and U-Net. We observe from this figure that i) the spectrum of the exact solution is very rich with representations of many frequencies, attesting to the *multi-scale* nature of the underlying problem and ii) there are significant differences in how CNO and FNO approximate the underlying solution in Fourier space. In particular, the decay in CNO’s spectrum is more accurate. On the other hand, the FNO spectrum is amplified along the horizontal axis, possibly on account of *aliasing errors* that add incorrect frequency content. The U-Net spectra are similar to that of CNO but with high-frequency modes being amplified, which leads to a higher test error. Next, in Figure 2 (right), we compare CNO, FNO and U-Net vis-a-vis the metric of *how the test error varies across resolutions*, see [SM C.4](#) for details, which is an important aspect for robust operator learning that been highlighted in [33, 25], see also [2] for a discussion with respect to representation equivalent neural operators or ReNOs. We find from Figure 2 (right) that for the Navier-Stokes benchmark, the FNO error is not *invariant* with respect to resolution, with an increase in error of up to 25% on lower-resolutions as well as a more modest but noticeable increase of 10% on resolutions, higher than the training resolution of 64<sup>2</sup>, implying that FNO is neither able to perform alias-error free super- or sub-resolution in this case, see also [2] for a detailed discussion on the resolution-invariance of FNO. Similarly, the increase of U-Net test error with respect to varying resolutions is even more pronounced, with a maximum increase of a factor of 3, indicating neither FNO nor U-Net are resolution (representation) equivalent in this case. In contrast, CNO error is invariant with respect to test resolution, verifying that it respects *continuous-discrete equivalence*. Further ablation studies for CNO are presented in [SM C.5](#).

**Efficiency.** In order to further compare CNO with FNO, which is the most widely used neural operator these days, we illustrate not just the performance in terms of errors but also terms in computational efficiency. To this end, in Figure 3 (left), we plot the size (total number of parameters) vs validation error for the Navier-Stokes benchmark for all FNO and CNO models that were considered in the model selection procedure to observe that for the same model size, CNO models led to significantly smaller validation errors. This resulted in smaller errors for the same per-epoch training time (Figure 3 (center)) for CNO vis-a-vis FNO. As observed in Figure 3 (center), this also implies that one of the best-performing CNO models, with a significantly smaller error than the best-performing FNO model is also almost twice as fast to train, thus showcasing the computational efficiency of CNOs.

**Scaling laws.** A very important aspect of modern deep learning is to evaluate how the performance of models scales with respect to data (and model size). To investigate this, we focus on the Navier-Stokes benchmark and present test error vs. (log of) number of training samples for GT, FNO and CNO in Figure 3 (right) to observe that the errors decrease consistently with number of samples. To quantify the rates of decrease, we fit **power laws** of the form  $E = (N_0/N)^{-r}$ , with  $E$  being the test error,  $N$  number of samples and  $N_0$  normalized to be the number of samples required to reach errors of 1% to obtain that CNO attains a much faster convergence rate ( $r = 0.37$ ) compared to FNO ( $r = 0.28$ ) and GT ( $r = 0.27$ ). This implies that CNO will require  $N_0 \approx 14.3K$  samples to attain 1% error which is 4-times less than FNO ( $N_0 \approx 60.1K$ ) and almost 10-times less than GT ( $N_0 \approx 133.2K$ ), highlighting the data efficiency of CNO. Finally, from Figure 3 (right), we also observe that a smaller CNO model (with 0.82 M parameters) scales worse (with  $r = 0.3$ ) compared to the best performing CNO model with 7.8M parameters, illustrating that model size could be a *bottleneck* for data scaling and larger models are required to scale with data.

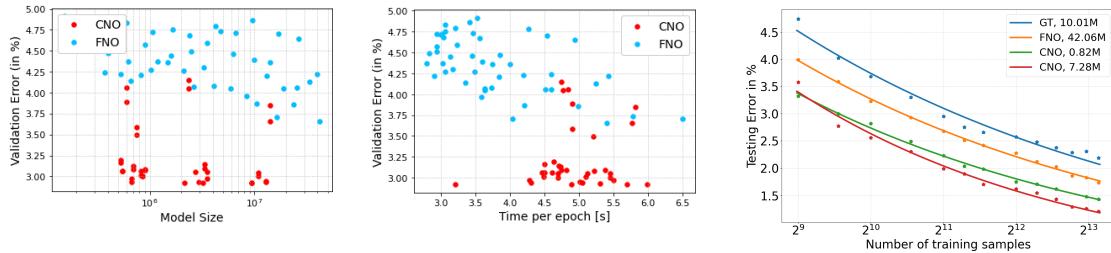


Figure 3: Navier-Stokes Benchmark: Test Error (Y-axis) vs. Model size (Left) and per-epoch training time (Center) for all the FNO and CNO models tested. The best-performing models are highlighted as is a small-scale yet efficient CNO model. Right: Log of Error vs. Log of Number of training samples for GT, FNO, CNO and a small-scale CNO model.

## 5 Discussion.

**Summary.** We propose CNO, a *novel* convolution-based architecture for learning operators. The basic design principle was to enforce a form of *continuous-discrete equivalence* in order to genuinely learn the underlying operators, rather than discrete representations of them. To this end, we modified the elementary operators of convolution, up-and downsampling and particularly nonlinear activations to realize CNO as a representation equivalent neural operator or ReNO in the sense of [2]. We also prove a universality theorem to show that CNO can approximate a large class of operators arising in PDEs to desired accuracy. A novel suite of experiments, termed as representative PDE benchmarks (**RPB**), encompassing a wide variety of PDEs, with multiple scales in the corresponding solutions, which are hard to resolve with traditional numerical methods, is also proposed and the model tested on them. We demonstrate that CNO outperforms the baselines, including FNO, significantly on most benchmarks. This also holds for the considered *out-of-distribution* testing tasks which ascertain the ability of the models to *generalize* to unseen data in a *zero-shot* manner.

**Comparison to Related Work.** We emphasize that our construction of CNO follows the theoretical prescription of recent paper [2] on enforcing structure preserving *continuous-discrete equivalence*. CNO is a *representation equivalent neural operator*, with respect to spaces of bandlimited functions, in the sense of [2]. Another motivating work for us is [22], see also [60], where the authors modify CNNs to eliminate (or reduce) aliasing errors in the context of image generation. We adapt the construction of [22] to our setting and deploy the resulting architecture in a very different context from that of [22], namely that of operator learning for PDEs rather than image generation. Moreover, we also instantiate CNO with a very different operator UNet architecture than that proposed in [22]. We would also like to mention related work on using CNNs for solving PDEs such as [1, 16] and emphasize that in contrast to CNO, they lack suitable notions of continuous-discrete equivalence. Finally comparing CNO to the widely used FNO model, we observe that unlike FNO which can fail to enforce CDE (see [14, 2] and Figure 2(right)), CNO preserves continuous-discrete equivalence. Moreover, the convolution operator in CNO is local in space, in contrast to convolution in Fourier space for FNO. The detailed empirical comparison presented here demonstrates CNO can outperform FNO and other baselines on many different metrics namely performance, computational efficiency, resolution invariance, out-of-distribution generalization as well as data scaling, paving the way for its widespread applications in science and engineering.

**Limitations and Future Work.** We have presented CNO for operators on an underlying two-dimensional Cartesian domain. The extension to three-space dimensions is conceptually straightforward but computationally demanding. Similarly, extending to non-Cartesian domains will require some form of transformation maps between domains, for instance reworking those suggested for FNO in [32, 57] can be readily considered. Adapting CNO to approximate trajectories (in time) of time-dependent PDEs, for instance by employing it in an *auto-regressive*

manner, is another possible extension of this paper. At the level of theoretical results, we believe that the generic framework of [11] can be adapted to show that not only does CNO approximate a large class of PDEs universally, it does so without incurring any curse of dimensionality, as shown for DeepONets in [26] and FNOs in [24]. Finally, adapting and testing CNO for learning operators, beyond the forward solution operator of PDEs is also interesting. One such direction lies in efficiently approximating *PDE inverse problems*, for instance those considered in [44].

**Supplementary Material for:**

Convolutional Neural Operators for Robust and Accurate Learning of PDEs.

**Table of Contents**

---

<b>A Technical Details for Section 2 of main text.</b>	<b>16</b>
A.1 Approximation of Operators mapping between Sobolev spaces by operators mapping between spaces of bandlimited functions . . . . .	16
A.2 Continuous-Discrete Equivalence for Operator $\mathcal{G}^*$ from Section 2.1 . . . . .	17
A.3 Multi-channel versions of elementary operators for CNO (2.3) . . . . .	19
A.4 Discrete operators for CNO . . . . .	20
A.5 Proof of Proposition 2.1 of Main Text . . . . .	22
<b>B Proof of Theorem 3.1 of Main Text</b>	<b>23</b>
B.1 Auxiliary results . . . . .	28
<b>C Technical Details for Section 4 of Main Text</b>	<b>29</b>
C.1 Training and Implementation Details . . . . .	29
C.2 Training Details . . . . .	36
C.3 Details about the description and numerical results in each benchmark . . . . .	37
C.4 Testing at Different Resolutions. . . . .	58
C.5 Ablation Studies. . . . .	60
C.6 Error vs. number of training samples. . . . .	61
<b>D Depiction of the Datasets.</b>	<b>61</b>

---

**A Technical Details for Section 2 of main text.****A.1 Approximation of Operators mapping between Sobolev spaces by operators mapping between spaces of bandlimited functions.**

We prove that one can approximate any continuous operator  $\mathcal{G}^\dagger : \mathcal{X} \rightarrow \mathcal{Y}$  (as introduced in Section 2) of the main text by an operator mapping between spaces of bandlimited functions to arbitrary accuracy. We obtain this result by discarding the high-frequency components, e.g. higher than frequency  $w$ , of both the input and output of  $\mathcal{G}^\dagger$ . This can be performed by a Fourier projection  $P_w$ . For orthogonal Fourier projections and also trigonometric polynomial interpolation [19, 24] the following result on the accuracy of the projection holds,

**Lemma A.1.** *Given  $\sigma, r \in \mathbb{N}_0$  with  $r > d/2$  and  $r \geq \sigma$ , and  $f \in C^r(\mathbb{T}^d)$  it holds for every  $w \in \mathbb{N}$  that,*

$$\|f - P_w(f)\|_{H^\sigma(\mathbb{T}^d)} \leq C(r, d) w^{-(r-\sigma)} \|f\|_{H^r(\mathbb{T}^d)}, \quad (\text{A.1})$$

for a constant  $C(r, d) > 0$  that only depends on  $r$  and  $d$ .

Using this result, we show that by discarding the high frequencies of the input and output of  $\mathcal{G}^\dagger$  one can approximate  $\mathcal{G}^\dagger$  to arbitrary accuracy by choosing an appropriate frequency cutoff.

**Lemma A.2.** *For any  $\varepsilon, B > 0$  there exist  $w \in \mathbb{N}$  such that  $\|\mathcal{G}^\dagger(a) - P_w \mathcal{G}^\dagger(P_w a)\|_{L^2(D)} \leq \varepsilon$  for all  $a \in H^r(D)$  with  $\|a\|_{H^r(D)} \leq B$ .*

*Proof.* We follow [24] and use Lemma A.1 repeatedly together with the stability of  $\mathcal{G}^\dagger$  (2.1) to obtain,

$$\begin{aligned} \|\mathcal{G}^\dagger(a) - P_w \mathcal{G}^\dagger(P_w a)\|_{L^2} &\leq \|\mathcal{G}^\dagger(a) - P_w \mathcal{G}^\dagger(a)\|_{L^2} + \|P_w \mathcal{G}^\dagger(a) - P_w \mathcal{G}^\dagger(P_w a)\|_{L^2} \\ &\lesssim w^{-r} \|\mathcal{G}^\dagger(a)\|_{H^r} + \|\mathcal{G}^\dagger(a) - \mathcal{G}^\dagger(P_w a)\|_{L^2} \\ &\lesssim w^{-r} \|\mathcal{G}^\dagger\|_{op} \|a\|_{H^r} + \omega(\|a - P_w a\|_{H^\sigma}) \\ &\lesssim w^{-r} \|\mathcal{G}^\dagger\|_{op} \|a\|_{H^r} + \omega(C w^{-(r-\sigma)} \|a\|_{H^r}). \end{aligned} \quad (\text{A.2})$$

It follows immediately that for large enough  $w$ ,

$$\sup_{\|a\|_{H^r} \leq B} \|\mathcal{G}^\dagger(a) - P_w \mathcal{G}^\dagger(P_w a)\|_{L^2} \leq \varepsilon. \quad (\text{A.3})$$

This proves the statement of the lemma.  $\square$

Given that both  $P_w a \in \mathcal{B}_w(D)$  and  $P_w \mathcal{G}^\dagger(P_w a) \in \mathcal{B}_w(D)$ , a consequence of the above lemma is the existence of an operator  $\mathcal{G}^* : \mathcal{B}_w(D) \rightarrow \mathcal{B}_w(D) : a \mapsto P_w \mathcal{G}^\dagger(a)$  that can approximate  $\mathcal{G}^\dagger$  arbitrarily well. It follows from the lemma and its proof that  $\|\mathcal{G}^\dagger - \mathcal{G}^*\|_{op} \leq \varepsilon$ , where the operators are considered as mappings from and to  $\mathcal{B}_w(D) \cap H^r(D)$  equipped with the  $H^r(D)$ -norm.

## A.2 Continuous-Discrete Equivalence for Operator $\mathcal{G}^*$ from Section 2.1

For every  $w > 0$ , we denote by  $\mathcal{B}_w(\mathbb{R}^2)$  the space of multivariate bandlimited functions

$$\mathcal{B}_w(\mathbb{R}^2) = \{f \in L^2(\mathbb{R}^2) : \text{supp } \widehat{f} \subseteq [-w, w]^2\},$$

where  $\widehat{f}$  denotes the Fourier transform on  $L^1(\mathbb{R})$

$$\widehat{f}(\xi) := \int_{\mathbb{R}} f(x) e^{-2\pi i x \xi} dx, \quad \xi \in \mathbb{R},$$

which extends to  $L^2(\mathbb{R})$  by a classical density argument. The set  $\Psi_w = \{\text{sinc}(2wx_1 - m) \cdot \text{sinc}(2wx_2 - n)\}_{m,n \in \mathbb{Z}}$  constitutes an orthonormal basis for  $\mathcal{B}_w(\mathbb{R}^2)$ . The bounded operator

$$T_{\Psi_w} : \ell^2(\mathbb{Z}^2) \rightarrow \mathcal{B}_w(\mathbb{R}^2), \quad T_{\Psi_w}(c_{m,n}) = \sum_{m,n \in \mathbb{Z}} c_{m,n} \text{sinc}(2w \cdot -m) \cdot \text{sinc}(2w \cdot -n),$$

which reconstructs a function from its basis coefficients, is called *synthesis operator*, and its adjoint

$$T_{\Psi_w}^* : \mathcal{B}_w(\mathbb{R}^2) \rightarrow \ell^2(\mathbb{Z}^2), \quad T_{\Psi_w}^* f = \left\{ f\left(\frac{m}{2w}, \frac{n}{2w}\right) \right\}_{m,n \in \mathbb{Z}},$$

which extract basis coefficients from an underlying function, is called *analysis operator*. Every bandlimited function can be uniquely and stably recovered from its sampled values  $\{f\left(\frac{m}{2w}, \frac{n}{2w}\right)\}_{m,n \in \mathbb{Z}}$  via the reconstruction formula

$$f(x_1, x_2) = T_{\Psi_w} T_{\Psi_w}^* f(x_1, x_2) = \sum_{m,n \in \mathbb{Z}} f\left(\frac{m}{2w}, \frac{n}{2w}\right) \text{sinc}(2wx_1 - m) \cdot \text{sinc}(2wx_2 - n), \quad (\text{A.4})$$

and we say that there is a *continuous-discrete equivalence (CDE)* between  $f$  and its samples  $\{f\left(\frac{m}{2w}, \frac{n}{2w}\right)\}_{m,n \in \mathbb{Z}}$ . More in general, every bandlimited function  $f \in \mathcal{B}_w(\mathbb{R}^2)$  can be uniquely and stably recovered from its values  $\{f(mT, nT)\}_{m,n \in \mathbb{Z}}$  if the *sampling rate* or reciprocal of grid size,  $1/T$  is greater or equal than the *Nyquist rate*  $2w$ . This simply follows from the fact that  $\mathcal{B}_w(\mathbb{R}^2) \subset \mathcal{B}_{w'}(\mathbb{R}^2)$  for every  $w' > w$ . On the contrary, reconstructing  $f \in \mathcal{B}_w$  at a sampling rate below the Nyquist rate, i.e.  $1/T < 2w$ , results in a non-zero value for the *aliasing error function*:

$$\varepsilon(f) = f - T_{\Psi_{\frac{1}{2T}}} T_{\Psi_{\frac{1}{2T}}}^* f,$$

and the associated *aliasing error*  $\|\varepsilon\|_2$  (cfr. Definition 2.1 in [2]).

Let  $\mathcal{G}^*$  be a (possibly) non-linear operator between band-limited spaces, i.e.  $\mathcal{G}^* : \mathcal{B}_w(\mathbb{R}^2) \rightarrow \mathcal{B}_{w'}(\mathbb{R}^2)$ , for some  $w, w' > 0$ . As argued in [2], the concepts of continuous-discrete equivalence (CDE) and aliasing error can be adapted to the operator  $\mathcal{G}^*$ . The continuous operator  $\mathcal{G}^*$  is uniquely determined by a map  $\mathbf{g}_{\Psi_w, \Psi_{w'}} : \ell^2(\mathbb{Z}^2) \rightarrow \ell^2(\mathbb{Z}^2)$  if the aliasing error operator

$$\varepsilon = \mathcal{G}^* - T_{\Psi_{w'}} \circ \mathbf{g}_{\Psi_w, \Psi_{w'}} \circ T_{\Psi_w}^* \quad (\text{A.5})$$

is identically zero, and we say that  $\mathcal{G}^*$  and  $\mathbf{g}_{\Psi_w, \Psi_{w'}}$  satisfy a continuous-discrete equivalence (cfr. Definition 3.1 in [2]). Equivalently, the diagram

$$\begin{array}{ccc} \mathcal{B}_w & \xrightarrow{\mathcal{G}^*} & \mathcal{B}_{w'}, \\ \downarrow T_{\Psi_w}^* & & \uparrow T_{\Psi_{w'}} \\ \ell^2(\mathbb{Z}^2) & \xrightarrow{\mathbf{g}_{\Psi_w, \Psi_{w'}}} & \ell^2(\mathbb{Z}^2) \end{array}$$

commutes, i.e. the black and the blue directed paths in the diagram lead to the same result. In this latter case, since  $T_{\Psi_w}^* \circ T_{\Psi_w}$  is the identity operator from  $\ell^2(\mathbb{Z}^2)$  onto itself, equation (A.5) forces the discretization  $\mathbf{g}_{\Psi_w, \Psi_{w'}}$  to be defined as

$$\mathbf{g}_{\Psi_w, \Psi_{w'}} = T_{\Psi_{w'}}^* \circ \mathcal{G}^* \circ T_{\Psi_w}, \quad (\text{A.6})$$

i.e. the diagram

$$\begin{array}{ccc} \mathcal{B}_w & \xrightarrow{\mathcal{G}^*} & \mathcal{B}_{w'} \\ T_{\Psi_w} \uparrow & & \downarrow T_{\Psi_{w'}}^* \\ \ell^2(\mathbb{Z}^2) & \xrightarrow{\mathfrak{g}_{\Psi_w, \Psi_{w'}}} & \ell^2(\mathbb{Z}^2) \end{array}$$

also commutes. In other words, once we fix the discrete representations associated to the input and output functions, there exists a unique way to define a discretization  $\mathfrak{g}_{\Psi_w, \Psi_{w'}}$  that is consistent with the continuous operator  $\mathcal{G}^*$  and this is given by (A.6). In practice, we may have access to different discrete representations of the input and output functions, e.g. point samples evaluated on different grids, which in the theory amounts to a change of reference systems in the function spaces. For instance, sampling a function  $f \in \mathcal{B}_w$  on a finer grid  $\left\{ \left( \frac{m}{2\bar{w}}, \frac{n}{2\bar{w}} \right) \right\}_{m,n \in \mathbb{Z}}$ ,  $\bar{w} > w$ , amounts to representing the function  $f$  with respect to the system  $\Psi_{\bar{w}} = \{ \text{sinc}(2\bar{w}x_1 - m) \cdot \text{sinc}(2\bar{w}x_2 - n) \}_{m,n \in \mathbb{Z}}$ , which constitutes an orthonormal basis for  $\mathcal{B}_{\bar{w}} \supset \mathcal{B}_w$ . Then, one can define the associated CDE discretization  $\mathfrak{g}_{\Psi_{\bar{w}}, \Psi_{\bar{w}'}}$  as in (A.6), and by equation (A.5), one readily obtains the change of basis formula

$$\mathfrak{g}_{\Psi_{\bar{w}}, \Psi_{\bar{w}'}} = T_{\Psi_{\bar{w}'}}^* \circ T_{\Psi_{w'}} \circ \mathfrak{g}_{\Psi_w, \Psi_{w'}} \circ T_{\Psi_w}^* \circ T_{\Psi_{\bar{w}}}, \quad (\text{A.7})$$

see also Remark 3.5 in [2] for a more general *change of frame* formula. Finally, all the above concepts generalize to every pair of frame sequences  $(\Psi, \Phi)$  that span respectively the input and output function spaces, and we refer to [2] for a complete exposition. Appendix A.2 can be adapted to bandlimited periodic functions, i.e. periodic functions with a finite number of non-zero Fourier coefficients, with the Dirichlet kernel as a counterpart of the sinc function, see [59, Section 5.5.2] for further details.

### A.3 Multi-channel versions of elementary operators for CNO (2.3)

In this section, we will define *multi-channel* versions of the elementary mappings which define CNO (2.3). Note that the single-channel versions were defined in the main text.

**Convolution Operator.** In the multi-channel settings, discrete kernels  $K_w$  are defined on the  $d_{in} \times d_{out} \times s^2$  uniform grids on  $D$ , where  $d_{in}$  is the number of input channels and  $d_{out}$  is the number of output channels. Formally, the kernels are defined as

$$K_{w,cl} = \sum_{i,j=1}^k k_{ij,cl} \cdot \delta_{z_{ij}}.$$

where  $c$  is the channel index in the input space, while  $l$  is the channel index in the output space. Each pair of channels defines corresponding single-channel convolution operation  $\mathcal{K}_{w,cl} : \mathcal{B}_w(D) \rightarrow \mathcal{B}_w(D)$ . For  $a \in \mathcal{B}_w(D, \mathbb{R}^{d_{in}})$ , the multi-channel convolution operation  $\mathcal{K}_w$  is defined as

$$(\mathcal{K}_w a(x))_l = \sum_{c=1}^{d_{in}} \mathcal{K}_{w,cl} a_c(x), \quad l = 1 \dots d_{out}.$$

**Upsampling and Downsampling Operators.** To upsample a signal  $a \in \mathcal{B}_w(D, \mathbb{R}^d)$  with  $d$  channels from the bandlimit  $w > 0$  to the bandlimit  $\bar{w} > w$ , one should apply the single-channel upsampling operator  $\mathcal{U}_{w, \bar{w}}$  to each individual channel of the input signal, independently. Formally, for  $a \in \mathcal{B}_w(D, \mathbb{R}^d)$ , the multi channel upsampling  $\mathcal{U}_{w, \bar{w}} : \mathcal{B}_w(D, \mathbb{R}^d) \rightarrow \mathcal{B}_{\bar{w}}(D, \mathbb{R}^d)$  is defined as

$$(\mathcal{U}_{w, \bar{w}} a(x))_c = a_c(x), \quad \forall x \in D, \quad c = 1 \dots d.$$

The downsampling operator of a signal  $a \in \mathcal{B}_w(D, \mathbb{R}^d)$  from the bandlimit  $w > 0$  to the bandlimit  $\underline{w} < w$  is defined in a similar manner (independent applications of the single-channel downsampling operators).

**Activation layer.** The multi-channel version of the activation layer, namely  $\Sigma_{w, \bar{w}} : \mathcal{B}_w(D, \mathbb{R}^d) \rightarrow \mathcal{B}_w(D, \mathbb{R}^d)$ , is realized by applying the single-channel activation layer to each of the  $d$  channels, independently.

#### A.4 Discrete operators for CNO

In this section, we will define the *discrete versions* of the elementary mappings in (2.3). Given a *discrete*, multi-channel signal  $a_s \in \mathbb{R}^{s \times s \times d}$  on  $s \times s \times d$  uniform grid, we will use the notation  $a_s[i, j, c]$  to refer to the  $(i, j)$ -th coordinate of the  $c$ -th channel of the signal, where  $i, j = 1 \dots s$  and  $c = 1 \dots d$ .

**Convolution operator.** Assume that instead of a continuous, single-channel signal  $a \in \mathcal{B}_w(D)$ , one has an access only to its sampled version  $a_s \in \mathbb{R}^{s \times s}$  on  $s \times s$  uniform grid on  $D$ . Assume that  $a_s$  is to be convolved with a *discrete* kernel  $K_w \in \mathbb{R}^{k \times k}$  with  $k = 2\hat{k} + 1$ . Let  $\hat{a}_s \in \mathbb{R}^{s+2\hat{k} \times s+2\hat{k}}$  be an extended version of  $a_s$  obtained by circular-padding or zero-padding of  $a_s$ . The discrete, single-channel convolution  $\mathcal{K}_s : \mathbb{R}^{s \times s} \rightarrow \mathbb{R}^{s \times s}$  of the signal  $a_s$  and the kernel  $K_w$  is given by

$$\mathcal{K}_s(a_s) = (a_s \star K_w)[i, j] = \sum_{m, n=-\hat{k}}^{\hat{k}} K_w[m, n] \cdot \hat{a}_s[i - m, j - n], \quad i, j = 1 \dots s,$$

where indices of  $\hat{a}_s$  outside the range  $1 \dots s$  correspond to the padded samples. By performing the convolution in a described way, we ensure that the input and the output signals have the same spatial dimension  $s \times s$ .

Let  $a_s \in \mathbb{R}^{s \times s \times d_{in}}$  be a discrete, multi-channel signal and  $K_w \in \mathbb{R}^{k \times k \times d_{in} \times d_{out}}$  a discrete kernel with  $k = 2\hat{k} + 1$ . The multi-channel convolution of  $a_s$  and  $K_w$  is defined by

$$(a_s \star K_w)[i, j, l] = \sum_{m, n=-\hat{k}}^{\hat{k}} \sum_{c=1}^{d_{in}} K_w[m, n, c, l] \cdot \hat{a}_s[i - m, j - n, c], \quad i, j = 1 \dots s,$$

where  $l$  corresponds to the index of the output channel and  $c$  to the index of the input channel.

**Upsampling and Downsampling Operators.** In this section, we will define the discrete upsampling and downsampling operators. For  $w > 0$ , let  $h_w$  be the interpolation *sinc* filter defined in 2.5. For a discrete, single-channel signal  $a_s \in \mathbb{R}^{s \times s}$ , let  $(\tilde{a}_s[n])_{n \in \mathbb{Z}}$  be its periodic extension into infinite length. In other words,  $\tilde{a}_s[n] = a_s[n \bmod s]$  for  $n \in \mathbb{Z}$ . The discrete upsampling  $\mathcal{U}_{s,N} : \mathbb{R}^{s \times s} \rightarrow \mathbb{R}^{Ns \times Ns}$  by an *integer factor*  $N \in \mathbb{N}$  of the signal  $a_s \in \mathbb{R}^{s \times s}$  is done in *two phases*:

1. First step is to increase the number of samples of the signal  $a_s$  from  $s^2$  to  $(Ns)^2$ . One transforms the signal  $a_s$  into the signal  $a_{s,\uparrow Ns}$  obtained by separating each two signal samples of  $a_s$  with  $N-1$  zero-valued samples. In other words, it holds that  $a_{s,\uparrow Ns} \in \mathbb{R}^{Ns \times Ns}$  and

$$a_{s,\uparrow Ns}[i,j] = \mathbb{1}_S(i) \cdot \mathbb{1}_S(j) \cdot a_s[i \bmod s, j \bmod s], \quad i, j = 1 \dots Ns,$$

where  $S = \{1, s+1, \dots, (N-1)s+1\}$  and  $\mathbb{1}_S$  is the indicator function.

2. Second step is to convolve the periodic extension of  $a_{s,\uparrow Ns}$  with the  $h_{s/2}$  interpolation filter to eliminate high frequency components. The upsampled signal is formally obtained by

$$\mathcal{U}_{s,N}(a_s)[i,j] = \sum_{n,m \in \mathbb{Z}} \tilde{a}_{s,\uparrow Ns}[n,m] \cdot h_{s/2}(is - ns, js - ms), \quad i, j = 1 \dots Ns.$$

The discrete downsampling  $\mathcal{D}_{s,N} : \mathbb{R}^{s \times s} \rightarrow \mathbb{R}^{s/N \times s/N}$  by an *integer factor*  $N \in \mathbb{N}$  of the signal  $a_s \in \mathbb{R}^{s \times s}$  is also done in *two phases* (under the assumption that  $s/N \in \mathbb{N}$ ):

1. First step is to convolve the periodic extension of  $a_s$  with the  $h_{s/(2N)}$  interpolation filter to eliminate high frequency content. Formally, the first step is defined by

$$a_{s,s/N}[i,j] = \sum_{n,m \in \mathbb{Z}} \tilde{a}_s[n,m] \cdot h_{s/(2N)}(is - ns, js - ms), \quad i, j = 1 \dots s/N.$$

2. Second step is to decrease the sampling rate of  $a_{s,s/N}$  by keeping *every*  $N$ -th sample of the signal. The downsampled signal is formally defined by

$$\mathcal{D}_{s,N}(a_s)[i,j] = a_{s,s/N}[(i-1)s+1, (j-1)s+1], \quad i, j = 1 \dots s/N.$$

Multi-channel discrete upsampling and downsampling are performed by independent applications of the corresponding single-channel operators.

Since perfect filters  $h_w$  have infinite impulse response and cause ringing artifacts around high-gradient points (e.g. discontinuities) due the Gibbs phenomenon, one usually uses *windowed-sinc* filters in the implementation. We will describe these filters later in the text (see C.1.4)

**Activation layer.** Given the definitions of the discrete operators, the discrete, single-channel activation layer is defined as

$$\Sigma_s : \mathbb{R}^{s \times s} \rightarrow \mathbb{R}^{s \times s}, \quad \Sigma_s(a_s) = \mathcal{D}_{s,N} \circ \sigma \circ \mathcal{U}_{s,N}(a_s),$$

where  $\sigma : \mathbb{R} \rightarrow \mathbb{R}$  is an activation function applied point-wise and  $N \in \mathbb{N}$  is a fixed constant. In our experiments, we noticed that  $N = 2$  is sufficient for accurate predictions. The multi-channel activation layer is performed by independent applications of the single-channel activation layer.

### A.5 Proof of Proposition 2.1 of Main Text

We use the same notation as in Section 2 and Appendix A.2. The layers of a convolutional neural operator (2.3) are given by,

$$v_{l+1} = \mathcal{P}_l \circ \Sigma_l \circ \mathcal{K}_l(v_l), \quad 0 \leq l \leq L-1, \quad (\text{A.8})$$

Hence, they consist of three elementary mappings between spaces of bandlimited functions, i.e.,  $\mathcal{K}_l$  is a convolution operator,  $\Sigma_l$  is a non-linear operator whose definition depends on the choice of an activation function  $\sigma : \mathbb{R} \rightarrow \mathbb{R}$ , and  $\mathcal{P}_l$  is a projection operator. We now show that CNO layers, whose discrete versions are outlined in the previous section, respect equation (A.6) and consequently CNOs are Representation equivalent Neural Operators (ReNOs) in the sense of [2, Definiton 3.4] and [2, Remark 3.5]. We recall that the convolutional operator appearing in (A.8) takes the form

$$\mathcal{K}_w f(x) = \sum_{m,n=-k}^k k_{m,n} f(x - z_{m,n}), \quad x \in \mathbb{R},$$

for some  $w > 0$ , where  $k \in \mathbb{N}$ ,  $k_{m,n} \in \mathbb{C}$  and  $z_{m,n} = \left\{ \left( \frac{m}{2w}, \frac{n}{2w} \right) \right\}_{m,n \in \mathbb{Z}}$ . By definition,  $\mathcal{K}_w$  is a well-defined operator from  $\mathcal{B}_w(\mathbb{R}^2)$  into itself. Moreover, its discretized version is defined by the mapping

$$\left\{ f \left( \frac{m}{2w}, \frac{n}{2w} \right) \right\}_{m,n \in \mathbb{Z}} \rightarrow \left\{ \mathcal{K}_w f \left( \frac{m}{2w}, \frac{n}{2w} \right) \right\}_{m,n \in \mathbb{Z}} = \left\{ \sum_{m',n'=-k}^k k_{m',n'} f(z_{m,n} - z_{m',n'}) \right\}_{m,n \in \mathbb{Z}},$$

and thus results in the commutative diagram

$$\begin{array}{ccc} \mathcal{B}_w & \xrightarrow{\mathcal{K}_w} & \mathcal{B}_w \\ T_{\Psi_w} \uparrow & & \downarrow T_{\Psi_w}^* \\ \ell^2(\mathbb{Z}^2) & \longrightarrow & \ell^2(\mathbb{Z}^2) \end{array}$$

Equivalently, the discretized verion of  $\mathcal{K}_w$  is defined via (A.6), which was to be shown. In order to define the activation layer  $\Sigma_l$ , we first assume that the activation function  $\sigma : \mathbb{R}^2 \rightarrow \mathbb{R}^2$  is such that for every  $f \in \mathcal{B}_w(\mathbb{R}^2)$

$$\sigma(f) \in \mathcal{B}_{\overline{w}}(\mathbb{R}^2), \quad (\text{A.9})$$

for some  $\bar{w} > w$ . In fact, in Section 2 we assume that the pointwise activation can be approximated by an operator between bandlimited spaces and consequently (A.9) is satisfied up to negligible frequencies. Thus, the activation layer  $\Sigma_{w,\bar{w}}: \mathcal{B}_w(\mathbb{R}^2) \rightarrow \mathcal{B}_{\bar{w}}(\mathbb{R}^2)$  in (A.8) is defined by the composition

$$\Sigma_{w,\bar{w}} = P_{\mathcal{B}_w(\mathbb{R}^2)} \circ \sigma \circ P_{\mathcal{B}_{\bar{w}}(\mathbb{R}^2)}, \quad (\text{A.10})$$

where  $P_{\mathcal{B}_w(\mathbb{R}^2)}: \mathcal{B}_{\bar{w}}(\mathbb{R}^2) \rightarrow \mathcal{B}_w(\mathbb{R}^2)$  denotes the orthogonal projection onto  $\mathcal{B}_w(\mathbb{R}^2)$  and  $P_{\mathcal{B}_{\bar{w}}(\mathbb{R}^2)}: \mathcal{B}_w(\mathbb{R}^2) \rightarrow \mathcal{B}_{\bar{w}}(\mathbb{R}^2)$  denotes the natural embedding of  $\mathcal{B}_w(\mathbb{R}^2)$  into  $\mathcal{B}_{\bar{w}}(\mathbb{R}^2)$ . The discretized version of each mapping in (A.10) is defined in order to guarantee a continuous-discrete equivalence (CDE) between the continuous and discrete levels. More precisely,  $P_{\mathcal{B}_w(\mathbb{R}^2)}$  and  $P_{\mathcal{B}_{\bar{w}}(\mathbb{R}^2)}$  are discretized via (A.6) as

$$\mathcal{D}_{\bar{w},w} = T_{\Psi_w}^* \circ P_{\mathcal{B}_w(\mathbb{R}^2)} \circ T_{\Psi_{\bar{w}}}, \quad \mathcal{U}_{w,\bar{w}} = T_{\Psi_{\bar{w}}}^* \circ P_{\mathcal{B}_{\bar{w}}(\mathbb{R}^2)} \circ T_{\Psi_w},$$

which are respectively called downsampling and upsampling. Consequently, the discretized version of the activation layer is given by the composition

$$\mathcal{D}_{\bar{w},w} \circ \sigma \circ \mathcal{U}_{w,\bar{w}},$$

which yields the commutative diagram

$$\begin{array}{ccccccc} \mathcal{B}_w & \xrightarrow{P_{\mathcal{B}_{\bar{w}}(\mathbb{R}^2)}} & \mathcal{B}_{\bar{w}} & \xrightarrow{\sigma} & \mathcal{B}_{\bar{w}} & \xrightarrow{P_{\mathcal{B}_w(\mathbb{R}^2)}} & \mathcal{B}_w \\ \uparrow T_{\Psi_w} & & \downarrow T_{\Psi_{\bar{w}}}^* & & \uparrow T_{\Psi_{\bar{w}}} & & \downarrow T_{\Psi_w}^* \\ \ell^2(\mathbb{Z}^2) & \xrightarrow{\mathcal{U}_{w,\bar{w}}} & \ell^2(\mathbb{Z}^2) & \xrightarrow{\sigma} & \ell^2(\mathbb{Z}^2) & \xrightarrow{\mathcal{D}_{\bar{w},w}} & \ell^2(\mathbb{Z}^2) \end{array}$$

which we wanted to show. Finally, the activation layer might be followed by an additional projective operator, i.e., by a downsampling or an upsampling. Thus, this exact correspondence between its constituent continuous and discrete operators establishes CNO as an example of Representation equivalent neural operators or ReNOs in the sense of [2, Definiton 3.4] and [2, Remark 3.5], thus proving Proposition 2.1 of the main text. As in Appendix A.2, the above proofs can be readily adapted to bandlimited periodic functions, i.e. periodic functions with a finite number of non-zero Fourier coefficients.

## B Proof of Theorem 3.1 of Main Text

We present the proof of a generalization of the universality result of Theorem 3.1. The theorem in the main text only holds when the differential operator  $\mathcal{L}$  only depends on the coordinate  $x$  through a coefficient function  $a \in H^r(D)$ . Although all benchmark PDEs in Section 4 satisfy this requirement, there are other important PDEs that do not, such as the standard elliptic PDE  $\nabla \cdot (a \nabla u) = f$ . We therefore generalize this requirement in the following setting,

**Setting B.1.** *We set  $D = \mathbb{T}^2$  and assume that the following is true,*

1.  $\mathcal{L}$  only depends on the coordinate  $x$  through functions  $a, f_1, \dots, f_\ell \in H^r(\mathbb{T}^2)$ .

2. The solution of the PDE characterized by  $a$  and  $f = (f_1, \dots, f_\ell)$  is given by a continuous operator  $\tilde{\mathcal{G}} : \tilde{\mathcal{X}} \subset (H^r(\mathbb{T}^2))^{\ell+1} \rightarrow H^r(\mathbb{T}^2) : (a, f) \mapsto u$  or  $u(T)$ , depending on the PDE. The operator of interest  $\mathcal{G}^\dagger$  is a restriction of  $\tilde{\mathcal{G}}$  for fixed  $f_1, \dots, f_\ell$  i.e.,  $\mathcal{G}^\dagger : \mathcal{X}^* \subset H^r(\mathbb{T}^d) \rightarrow H^r(\mathbb{T}^d) : a \mapsto \tilde{\mathcal{G}}(a, f_1, \dots, f_\ell)$ .
3. Similar to (2.1), it holds for all  $(a, f), (a', f') \in \mathcal{X}^*$  it holds that

$$\left\| \tilde{\mathcal{G}}(a, f) - \tilde{\mathcal{G}}(a', f') \right\|_{L^p(\mathbb{T}^2)} \leq \omega \left( \|a - a'\|_{H^\sigma(\mathbb{T}^2)} + \max_i \|f_i - f'_i\|_{H^\sigma(\mathbb{T}^2)} \right), \quad (\text{B.1})$$

for some  $p \in \{2, \infty\}$  and  $\sigma \in \mathbb{N}_0$  with  $\sigma < r$ . This is automatically satisfied if  $\mathcal{X}^*$  is compact and  $\tilde{\mathcal{G}}$  is continuous [24].

4. It holds that the activation function  $\sigma$  is at least  $r$  times continuously differentiable and not a polynomial. (See Remark B.4 for a generalization.)

In addition, we will use the following notation in the proof.

- For  $J \in \mathbb{N}$  we define for every  $j \in \{0, \dots, J-1\}^2$  the grid  $\mathbf{x}_j^J = (2\pi j_1/J, 2\pi j_2/J)$ .
- We denote the Fourier basis by  $\{\mathbf{e}_\mathbf{k}\}_{\mathbf{k} \in \mathbb{Z}^2}$ , following the notation of [24]. For  $\mathbf{k} = (\mathbf{k}_1, \dots, \mathbf{k}_d) \in \mathbb{Z}^d$ , we let  $\sigma(\mathbf{k})$  be the sign of the first non-zero component of  $\mathbf{k}$  and we define

$$\mathbf{e}_\mathbf{k} := C_\mathbf{k} \begin{cases} 1, & \sigma(\mathbf{k}) = 0, \\ \cos(\mathbf{k} \cdot \mathbf{x}), & \sigma(\mathbf{k}) = 1, \\ \sin(\mathbf{k} \cdot \mathbf{x}), & \sigma(\mathbf{k}) = -1, \end{cases} \quad (\text{B.2})$$

where the factor  $C_\mathbf{k} > 0$  ensures that  $\mathbf{e}_\mathbf{k}$  is properly normalized, i.e. that  $\|\mathbf{e}_\mathbf{k}\|_{L^2(\mathbb{T}^d)} = 1$ .

- For  $N \in \mathbb{N}$  let  $P_N$  denote a trigonometric polynomial interpolation operator as in (B.16) in **SM B.1**.

Assuming Setting B.1 we can now prove the following theorem on the universality of CNOs. In the proof we will construct an operator  $\mathcal{G} : H^r(\mathbb{T}^2) \rightarrow C(\mathbb{T}^2)$ , mapping between function spaces, and we will therefore allow to apply the activation function to the continuous representation of the signal rather than an upsampled version. We then make the link to the discrete implementation of the CNO by considering an encoder  $\mathcal{E}_K$  that maps the input function  $a$  to the evaluation of  $a$  on a grid, enhanced by some Fourier features [55] in case  $\ell > 0$  in Setting B.1.

**Theorem B.2.** Let  $\sigma \in \mathbb{N}_0$  and  $p \in \{2, \infty\}$  as in (B.1),  $r > \max\{\sigma, 2/p\}$  and  $B > 0$ . For any  $\varepsilon > 0$  and any operator  $\mathcal{G}^\dagger$  satisfying Setting B.1, there exist  $K, N \in \mathbb{N}_0$  and a CNO  $\mathcal{G} : H^r(\mathbb{T}^2) \rightarrow C(\mathbb{T}^2)$  such that for every  $a \in \mathcal{X}^*$  with  $\|a\|_{H^r(\mathbb{T}^2)} \leq B$  it holds,

$$\left\| \mathcal{G}^\dagger(a) - \mathcal{G}(a) \right\|_{L^p(\mathbb{T}^2)} < \varepsilon. \quad (\text{B.3})$$

The CNO is implemented through an encoder

$$\mathcal{E}_K : H^r(\mathbb{T}^2) \rightarrow (\mathbb{R}^{N \times N})^{(K+1)^2} : a \mapsto (a(\mathbf{x}^N), (\cos(\mathbf{k} \cdot \mathbf{x}^N), \sin(\mathbf{k} \cdot \mathbf{x}^N))_{1 \leq \|\mathbf{k}\|_\infty \leq K}) \quad (\text{B.4})$$

and a single invariant block  $\widehat{\Phi} : (\mathbb{R}^{N \times N})^{(K+1)^2} \rightarrow \mathbb{R}^{N \times N}$  such that  $\mathcal{G}(a)(\mathbf{x}^N) = (\widehat{\Phi} \circ \mathcal{E}_K)(a)$ . If  $\ell = 0$  (see Setting B.1) then  $K = 0$ , meaning that no Fourier features are needed.

*Proof.* Let  $M, N \in \mathbb{N}$  with  $N/M \in \mathbb{N}$ . We will construct a CNO with input  $a(\mathbf{x}^N)$  and the Fourier features  $\mathbf{e}_k(\mathbf{x}^N)$  for  $\mathbf{k} \in \mathcal{K} := \{-M/2, -M/2 + 1, \dots, M/2\}^2 \setminus \{0, 0\}$ , summarized in the tensor  $(a(\mathbf{x}^N), \mathbf{e}^{M/2}(\mathbf{x}^N)) := (a(\mathbf{x}^N), (\mathbf{e}_k(\mathbf{x}^N))_{\mathbf{k} \in \mathcal{K}})$ . In the proof, we will use the property that bandlimited functions can be represented by their function values on a fine enough grid. We will therefore first construct a continuous operator  $\mathcal{G} : H^r(\mathbb{T}^2) \rightarrow C(\mathbb{T}^2)$  that is a good approximation of  $\mathcal{G}^\dagger$ . In the second step, we will then prove that  $\mathcal{G}(a)(\mathbf{x}^N)$  indeed corresponds to a CNO.

**Step 1: construction of  $\mathcal{G}$ .** First, since  $a, f \in H^r(\mathbb{T}^2)$  we can use Lemma A.1 and assumption (B.1) on the stability of  $\widetilde{\mathcal{G}}$  to find that,

$$\left\| \widetilde{\mathcal{G}}(a, f) - \widetilde{\mathcal{G}}(P_M(a, f)) \right\|_{L^p(\mathbb{T}^2)} \leq \omega \left( C_{B,f} M^{-(r-\sigma)} \right). \quad (\text{B.5})$$

Next, we define for any  $J \in \mathbb{N}$  the set

$$\mathcal{A}_J = \{ \mathbf{y} \in (\mathbb{R}^{J \times J})^{(M+1)^2} \mid \exists a \in H^r(\mathbb{T}^2) : \mathbf{y} = (a(\mathbf{x}^J), \mathbf{e}^{M/2}(\mathbf{x}^J)) \text{ and } \|a\|_{H^r(\mathbb{T}^2)} \leq B \}, \quad (\text{B.6})$$

and the map,

$$G : \mathcal{A}_M \subset (\mathbb{R}^{M \times M})^{(M+1)^2} \rightarrow \mathbb{R} : (a(\mathbf{x}^M), \mathbf{e}^{M/2}(\mathbf{x}^M)) \mapsto \widetilde{\mathcal{G}}(P_M(a, f))(\mathbf{x}_{0,0}). \quad (\text{B.7})$$

The existence of the map  $G$  can be justified as follows. Let  $P_M$  denote a trigonometric polynomial interpolation operator as in (B.16) in SM B.1. By the Nyquist–Shannon sampling theorem and the Whittaker–Shannon interpolation formula there is a bijection between the discrete values  $(a(\mathbf{x}^M), \mathbf{e}^{M/2}(\mathbf{x}^M))$  and  $P_M a$  and  $\mathbf{e}^{M/2}$ , and therefore also  $P_M a$  and  $P_M f_i$  for all  $1 \leq i \leq \ell$ . Hence, the mapping  $G$  is equivalent to the mapping  $P_M(a, f) \mapsto P_M u$ , and therefore well-defined. The continuity of  $G$  follows from that of  $\widetilde{\mathcal{G}}$ . By the universal approximation theorem (Theorem B.6) there exists a shallow neural network  $\Psi$  such that  $|\Psi(\mathbf{y}) - G(\mathbf{y})| < \varepsilon$  for all  $\mathbf{y} \in \mathcal{A}_M$ . Note that  $\Psi$  only provides an approximation in the point  $\mathbf{x}_{0,0}$ . We can expand  $\Phi$  to the whole  $\mathbb{T}^2$  by defining the operator  $\Psi^*$  as follows,

$$\Psi^* : \mathcal{X}^* \rightarrow C(\mathbb{T}^2) : a \mapsto \left[ \mathbb{T}^2 \ni \mathbf{z} \mapsto \Psi \left( a(\mathbf{z} + \mathbf{x}^M), \mathbf{e}^{M/2}(\mathbf{z} + \mathbf{x}^M) \right) \right]. \quad (\text{B.8})$$

For the intuition of the reader: the extension from  $\Psi$  to  $\Psi^*$  is similar to the extension from the local stencil of a finite difference scheme to its corresponding global approximation. As a result,  $\Psi^*$  has the same accuracy as  $\Psi$ ,

$$\left\| \mathcal{G}^\dagger(P_M a) - \Psi^*(P_M a) \right\|_{C^0(\mathbb{T}^2)} < \varepsilon. \quad (\text{B.9})$$

We finalize our construction by projecting  $\Psi^*(P_M a)$  on to the space of trigonometric polynomials. The accuracy of such a projection is given by Lemma A.1,

$$\|(P_N - \text{Id})\Psi^*(a)\|_{L^p(\mathbb{T}^2)} \leq \|(P_N - \text{Id})\Psi^*(a)\|_{H^{1-2/p}(\mathbb{T}^2)} \leq CN^{-(r-2/p)} \|\Psi^*(a)\|_{H^r(\mathbb{T}^2)}, \quad (\text{B.10})$$

where we used that either  $p = 2$  or  $p = \infty$ . It is important to note that  $\|\Psi^*(a)\|_{H^r(\mathbb{T}^2)}$  is independent of  $N$ . We then define the operator  $\mathcal{G}$  as,

$$\mathcal{G}(a)(\mathbf{z}) = (P_N \circ \Psi^*)(P_M a)(\mathbf{z}). \quad (\text{B.11})$$

Finally, we can put all obtained estimates together to find,

$$\begin{aligned} & \left\| \mathcal{G}^\dagger(a) - \mathcal{G}(a) \right\|_{L^p(\mathbb{T}^2)} \\ & \leq \left\| \mathcal{G}^\dagger(a) - \mathcal{G}^\dagger(P_M a) \right\|_{L^p(\mathbb{T}^2)} + \left\| \mathcal{G}^\dagger(P_M a) - \Psi^*(P_M a) \right\|_{C^0(\mathbb{T}^2)} \\ & \quad + \left\| \Psi^*(P_M a) - \mathcal{G}(a) \right\|_{L^p(\mathbb{T}^2)} \\ & \leq \omega \left( C_{B,f} M^{-(r-\sigma)} \right) + \varepsilon + C_{B,M,\varepsilon} N^{-(r-2/p)}. \end{aligned} \quad (\text{B.12})$$

It then follows that one can make this upper bound arbitrarily small by choosing  $\varepsilon$  sufficiently small and  $M, N$  sufficiently large (in that order).

**Step 2:  $\mathcal{G}$  corresponds to a CNO.** We will now show that the operator  $\mathcal{G}$  is in agreement with our definition of a convolutional neural operator (CNO). To do so, we will use that trigonometric polynomials up to a certain degree can be exactly retrieved based on their values on a grid (see SM B.1 and [19]).

First of all, given that  $N/M \in \mathbb{N}$  we find that the functions  $P_M a$  and  $\mathbf{e}^{M/2}$  can be exactly recovered from their discrete values on the grid  $\mathbf{x}^N$ . We therefore will look for a CNO with input  $\mathbf{y} = (a(\mathbf{x}^N), \mathbf{e}^{M/2}(\mathbf{x}^N)) \in \mathcal{A}_N$  for which the continuous representation of the output agrees with  $\mathcal{G}(a)$ .

A crucial next observation is that  $G$  is equivariant with respect to translations in space of the input (simultaneously across all channels). By [46, Theorem 1.1] there then exists a shallow CNN  $\Phi$  such that  $\pi \circ \Phi = \Psi$ , where  $\pi$  is the projection on the first coordinate  $\pi : (\mathbb{R}^{N \times N})^{(M+1)^2} \rightarrow \mathbb{R}^{(M+1)^2} : X \mapsto (X_{1,1}^1, \dots, X_{1,1}^\ell)$  (as in [46]). For simplicity we will assume that the CNN is of the form  $\Phi(\mathbf{y}) = K_2 * \sigma(K_1 * \mathbf{y})$ , i.e. that it only has one channel at every layer. The proof of the general case is completely analogous, but much heavier on notation.

We then lift the convolution filter  $K_1 \in \mathbb{R}^{M \times M}$  to the grid  $\mathbb{R}^{N \times N}$  by using a stride of  $N/M$  and filling up the rest by zeroes. More rigorously, we consider the matrix  $\widehat{K}_1 := K_1 \otimes E$  with  $E_{ij} = \delta_{i1}\delta_{j1}$ . Similarly we define  $\widehat{K}_2 := K_2 \otimes E$ . We can then define a new CNN  $\widehat{\Phi} : \mathcal{A}_N \rightarrow \mathbb{R}^{N \times N} : \mathbf{y} \mapsto \widehat{K}_2 * \sigma(\widehat{K}_1 * \mathbf{y})$ . The output of  $\widehat{\Phi}$  then consists of approximations of  $\mathcal{G}^\dagger(a)$  at  $(N/M)^2$  different  $M \times M$  subgrids of  $\mathbf{x}^N$ , i.e. all possible translations of  $\mathbf{x}^M$  within  $\mathbf{x}^N$ . More precisely, it holds that

$$\Psi^*(\mathbf{x}^N) = \widehat{\Phi} \left( P_M a(\mathbf{x}^N), \mathbf{e}^{M/2}(\mathbf{x}^N) \right) \in \mathbb{R}^{N \times N}. \quad (\text{B.13})$$

Moreover, since the operator  $P_N$  only uses the values of  $\Psi^*$  on the grid  $\mathbf{x}^N$  it follows that applying  $P_N$  to the right-hand side of the above equation or applying an interpolation sinc filter with corresponding frequency gives the exact same result,

$$\mathcal{G}(a)(\mathbf{x}^N) = (P_N \circ \Psi^*)(P_M a)(\mathbf{x}^N) = h_N \star \widehat{\Phi} \left( P_M a(\mathbf{x}^N), \mathbf{e}^{M/2}(\mathbf{x}^N) \right). \quad (\text{B.14})$$

The right-hand side exactly corresponds to our definition of a CNO, thereby concluding the universality proof.  $\square$

**Remark B.3** (Alternative proof). *We stress that it is crucial in the proof that  $M$  can be chosen independently of  $N$ . A straightforward application of [46, Theorem 1.1] on the map  $G$  with  $N = M$  would not lead to an accurate CNO approximation as the  $\|\Psi^*(a)\|_{H^r(\mathbb{T}^2)}$  will depend on  $N = M$  such that  $N^{-(r-2/p)} \|\Psi^*(a)\|_{H^r(\mathbb{T}^2)}$  might not converge to zero. In addition, because of the used trick we obtain convolution kernels with a stride of  $N/M$  and therefore a sparse kernel. An alternative strategy could be to replace the universal approximation (Theorem B.6) by an approximation theorem that provides explicit control on the network size and upper bounds on the weights such as those in [10]. Other than a much more complicated proof, one will also need to put stronger regularity conditions on  $\mathcal{G}^\dagger$ .*

**Remark B.4** (Polynomial and rational activations). *The CNO constructed in the above theorem is exactly equivariant. As suggested in [22], it can be sufficient in practice to break this perfect equivariance by applying the activation function  $\sigma$  to an upsampled discrete version of the signal rather than the continuous representation of the signal. We do the same in our implementation of CNO. Note that if one would use polynomial activation functions one could still recover exact equivariance by choosing a high enough upsampling rate. In this case the universality of CNOs can be proven by replacing the universal approximation theorem for neural networks (Theorem B.6) by the Weierstrass approximation theorem. The rest of the proof of Theorem B.2 remains unchanged. Similarly, one could consider using Padé and rational approximants as activation functions [56, 43, 12, 6]. The computation of a rational activation function  $\sigma(x) = p(x)/q(x)$  can then be approximated by iteratively minimizing  $\|p(x)\sigma(x) - q(x)\|_2^2$ , following the idea of [62]. Methods such as Newton-Raphson only involve multiplications and can therefore be completely applied in an alias-free way through proper upsampling before, and downsampling after each multiplication.*

**Remark B.5** (Physics-informed CNOs). *Physics-informed learning employs a PDE residual-based loss to circumvent the need for training data. Examples of such frameworks include physics-informed neural networks (PINNs) [49], physics-informed DeepONets [61] and physics-informed FNOs [36]. Using the continuous representation of the CNO output, one can use automatic differentiation to create a physics-informed CNO loss. In order to use the tools provided in [11, Theorem 3.9] to obtain a bound on the approximation error for physics-informed CNOs, one needs to prove that the CNO error converges at a certain rate in terms of its size. Although Theorem B.2 does not provide such a rate, its proof does give a hint of which stronger assumptions are needed to obtain this result. The most notable ingredients include a stronger stability result (B.1) with a continuity modulus  $\omega$  decreasing at least at a polynomial rate, and a stronger regularity assumption on  $G$  (B.7), and hence  $\mathcal{G}^\dagger$ .*

## B.1 Auxiliary results

We list the auxiliary results that are used in the proof of Theorem B.2. First, we state a well-known version of the universal approximation theorem for feedforward neural networks [31]:

**Theorem B.6** ([31]). *Let  $\sigma : \mathbb{R} \rightarrow \mathbb{R}$  be a function that is locally essentially bounded with the property that the closure of the set of points of discontinuity has zero Lebesgue measure. For  $1 \leq j \leq n$ , let  $\alpha_j, \theta_j \in \mathbb{R}$  and  $y_j \in \mathbb{R}^d$ . Then finite sums of the form*

$$G(x) = \sum_{j=1}^N \alpha_j \sigma(y_j^T x + \theta_j), \quad x \in \mathbb{R}^d \quad (\text{B.15})$$

*are dense in  $C(\mathbb{R}^d)$  if and only if  $\sigma$  is not an algebraic polynomial.*

Next, we demonstrate the equivalence of using the interpolation sinc filter (2.5) and trigonometric polynomial interpolation. If you sample a function  $f \in C(\mathbb{T} = [0, 2\pi))$  with sampling frequency  $\frac{2\pi}{N}$ , the result obtained through trigonometric polynomial interpolation  $P_N f$  is given by [19],

$$P_N f(x) = \begin{cases} \sum_{|n| \leq (N-1)/2} \frac{1}{N} \sum_{j=0}^{N-1} f(x_j) \exp(in(x - x_j)), & \text{for } n \text{ odd}, \\ \sum_{|n| \leq N/2} \frac{1}{N c_n} \sum_{j=0}^{N-1} f(x_j) \exp(in(x - x_j)), & \text{for } n \text{ even}, \end{cases} \quad (\text{B.16})$$

where  $x_j = 2\pi j/N$ ,  $c_n = 1$  for  $|n| < N/2$  and  $c_n = 2$  for  $|n| = N/2$ . We will prove that one obtains the exact same result by using an interpolation sinc filter with the same frequency on the periodic extension of  $f$ . We prove this result in the one-dimensional case for odd  $N$ . The result for even  $N$  follows in an identical way, the result for the multi-dimensional case through tensorisation.

**Lemma B.7.** *For any  $N \in 2\mathbb{N} + 1$  and  $f \in C(\mathbb{T})$  it holds that,*

$$P_N f(x) = \frac{1}{N} \sum_{|n| \leq (N-1)/2} \sum_{j=0}^{N-1} f(x_j) \exp(in(x - x_j)) = \sum_{n \in \mathbb{Z}} f(x_n) \operatorname{sinc}\left(N \cdot \frac{x - x_n}{2\pi}\right) \quad (\text{B.17})$$

*Proof.* As a first step, it follows from [19, Section 2.2.2] that

$$P_N f(x) = \frac{1}{N} \sum_{|n| \leq (N-1)/2} \sum_{j=0}^{N-1} f(x_j) \exp(in(x - x_j)) = \frac{1}{N} \sum_{n=0}^{N-1} f(x_n) \frac{\sin(N(x - x_n)/2)}{\sin((x - x_n)/2)}. \quad (\text{B.18})$$

Then we use the result from [51], where we replace their  $N$ -periodic signal  $x(t)$  by the function  $f(x)$  through the transformation  $t = Nx/2\pi$  and  $x(t) = f(2\pi t/N)$ . In their notation, but with the change that here we use the *normalized* sinc function ( $\operatorname{sinc}(x) = \sin(\pi x)/\pi x$  for  $x \neq 0$ ), [51] shows that

$$\sum_{n \in \mathbb{Z}} x(n) \operatorname{sinc}(t - n) = \frac{\sin(\pi t)}{N} \sum_{n=-L}^{M-1} x(n) (-1)^n \csc(\pi(t - n)/N) \quad (\text{B.19})$$

with  $L, M \in \mathbb{N}_0$  such that  $L + M = N$ . We will take  $L = 0$  and  $M = N$ , and use that  $\csc(z) = 1/\sin(z)$  and that  $\cos(\pi n) = (-1)^n$  and  $\sin(\pi n) = 0$  to obtain,

$$\sum_{n \in \mathbb{Z}} x(n) \operatorname{sinc}(t - n) = \frac{1}{N} \sum_{n=-L}^{M-1} x(n) \frac{\sin(\pi(t - n))}{\sin(\pi(t - n)/N)}, \quad (\text{B.20})$$

which is equivalent to,

$$\sum_{n \in \mathbb{Z}} f(x_n) \operatorname{sinc}(N(x - x_n)/2\pi) = \frac{1}{N} \sum_{n=0}^{N-1} f(x_n) \frac{\sin(N(x - x_n)/2)}{\sin((x - x_n)/2)}. \quad (\text{B.21})$$

Combining all obtained equalities proves the claim.  $\square$

## C Technical Details for Section 4 of Main Text

### C.1 Training and Implementation Details

We start with a succinct description of the baselines that were used in the main text.

#### C.1.1 Feed Forward Dense Neural Networks

Given an input  $u \in \mathbb{R}^m$ , a feedforward neural network (also termed as a multi-layer perceptron), transforms it to an output, through a layer of units (neurons) which compose of either affine-linear maps between units (in successive layers) or scalar nonlinear activation functions within units [15], resulting in the representation,

$$\bar{u}_\theta(y) = C_L \circ \sigma \circ C_{L-1} \dots \circ \sigma \circ C_2 \circ \sigma \circ C_1(u). \quad (\text{C.1})$$

Here,  $\circ$  refers to the composition of functions and  $\sigma$  is a scalar (nonlinear) activation function. For any  $1 \leq \ell \leq L$ , we define

$$C_\ell z_\ell = W_\ell z_\ell + b_\ell, \text{ for } W_\ell \in \mathbb{R}^{d_{\ell+1} \times d_\ell}, z_\ell \in \mathbb{R}^{d_\ell}, b_\ell \in \mathbb{R}^{d_{\ell+1}}, \quad (\text{C.2})$$

and denote,

$$\theta = \{W_\ell, b_\ell\}_{\ell=1}^L, \quad (\text{C.3})$$

to be the concatenated set of (tunable) weights for the network. Thus in the terminology of machine learning, a feed forward neural network (C.1) consists of an input layer, an output layer, and  $L$  hidden layers with  $d_\ell$  neurons,  $1 < \ell < L$ . In all numerical experiments, we consider a uniform number of neurons across all the layer of the network  $d_\ell = d_{\ell-1} = d$ ,  $1 < \ell < L$ . The first baseline model consists into a feed forward neural network with *residual blocks* which use *skip or shortcut connections* [18]. A residual block spanning  $k$  layers is defined as follows,

$$r(z_\ell, z_{\ell-k}) = \sigma(W_\ell z_\ell + b_\ell) + z_{\ell-k}. \quad (\text{C.4})$$

The residual network takes as input a sample function  $u \in \mathcal{X}$ , encoded at  $m = s \times s$  *Cartesian grid* points  $(x_1, \dots, x_m)$ ,  $\mathcal{E}(u) = (u(x_1), \dots, u(x_m)) \in \mathbb{R}^m$ , and outputs the output sample  $\mathcal{G}(u) \in \mathcal{Y}$  encoded at the same set of points,  $\mathcal{E}(\mathcal{G}(u)) = (\mathcal{G}(u)(x_1), \dots, \mathcal{G}(u)(x_m)) \in \mathbb{R}^m$ . In all the experiments, but the compressible Euler,  $s = 64$ . Instead, for the compressible Euler equation, the sampling rate is  $s = 128$ . The number of layers  $L$ , neurons  $d$  are chosen through cross-validation, whereas the activation function  $\sigma$  corresponds to a Leaky ReLU and the depth of the residual block  $k$  is fixed and equal to 2.

### C.1.2 ResNet

For the ResNet baseline, we adopt a convolutional neural network architecture with additional skip connections, as described in [18]. The architecture begins with an initial block composed of a convolutional layer with a  $7 \times 7$  kernel, zero padding, and a ReLU activation function, all followed by batch normalization. The first layer generates an output with a channel count of  $c$ . Subsequently, the output from the initial block undergoes downsampling via a second block. This second block consists of two sub-blocks, each mirroring the structure of the initial block but with a smaller  $3 \times 3$  convolutional layer, a stride of 2, and padding of 1. The channel count doubles within each of these sub-blocks.

The downsampled output is then processed through a series of  $N_{res}$  residual blocks, as defined in equation C.4. Each residual block consists of a convolution operation, batch normalization, and ReLU activation.

Finally, the signal is upsampled through a pair of blocks comprising transposed convolution, batch normalization, and ReLU activation.

The complete architecture is available at repository of the paper [20]:<https://github.com/junyanz/pytorch-CycleGAN-and-pix2pix/blob/master/models/networks.py>.

### C.1.3 UNet

For the UNet baseline we use the model architecture proposed in [50]. However, we slightly modify the proposed architecture by varying the number of output channels  $c$  of the first convolutional layer, which is chosen through cross validation. We ensure that the number of channels used in the subsequent layers align with the chosen value of  $c$ . Specifically, we respect the progressive increase or decrease in the number of channels as established in the original architecture across different layers.

### C.1.4 Convolutional Neural Operator

**Design of the filters.** As we noted before, perfect *sinc* interpolation filters  $h_w$  have infinite impulse response and cause ringing artifacts around high-gradient points due the Gibbs phenomenon. In practice, one uses *windowed-sinc* filters which serve as convenient approximations of  $h_w$ . They have finite impulse response and weakened ringing effect [59].

The *windowed-sinc* filters are constructed by multiplying the ideal filter  $h_w$  by a corresponding *window function* of *finite* length. That is equivalent to convolving the filter with the window function in the frequency domain. To design the windowed filter, one can use standard Python libraries and their functions such as `scipy.signal.firwin`. By using this function, we are enabled to manually control the cutoff frequency  $w_c$  and the half-width of the transition band  $w_h$  of the designed filters. We design discrete filters with a prescribed compact support  $N_{tap} \in \mathbb{N}$ . In this case, we say that a designed filter has  $N_{tap}$  *taps*. Implementation of the filters is borrowed from [22] (CUDA programming model).

We show several *1D* designed filters in the Figure 4, where we set  $w_c = s/(2 + \varepsilon)$ , for  $\varepsilon \ll 1$ . We control the half-width of the filter  $w_h = c_h \cdot s$  by controlling the coefficient  $c_h$ . When  $c_h$  is set to 0.5, one would anticipate the design of an almost perfect *sinc* filter. However, the presence of undesirable oscillations in the frequency domain can be observed due to the finite impulse response of windowed filters, as depicted in Figure 4. That is why we set  $c_h$  to be at least 0.6. One can implement a *2D* filter by first convolving a *1D* filter with each row and then with each column.

The activation layer  $\Sigma_{w, \bar{w}}$  plays a vital role in the CNO model. It is essential to closely examine the ratio  $N_\sigma = \bar{w}/w$  as a significant parameter. To facilitate implementation of the CNO, we make the assumption that  $N_\sigma \in \mathbb{N}$  and  $N_\sigma \geq 2$ . Throughout the entire architecture, we make the assumption that the value of  $N_\sigma$  remains fixed. In our implementation of the CNO, it is worth noting that the value of  $N_\sigma$  can also be a rational number if the sampling rate of an input signal requires it (e.g. if one wants to upsample a signal from the sampling rate 11 to the sampling rate 20).

We choose to fix the coefficient  $w_c = s/2.0001$ , so that the cutoff frequency is very close to the *Nyquist critical frequency*. It remains to choose the number of taps  $N_{tap}$ , the coefficient related to the half-width of the filter  $c_h$  and the ratio related to the activation layer  $N_\sigma$ .

**Choice of parameters.** Throughout our experiments, we maintained a consistent configuration, setting  $N_\sigma$  to 2,  $c_h$  to 0.8, and  $N_{tap}$  to 12. Prior to finalizing the filter parameters, we conducted experiments using various filters; however, no significant differences were observed. To further validate this assumption, we conducted the Navier-Stokes experiment using different filter designs. First, we selected the best-performing CNO model based on the criteria described in C.2 using filter parameters  $N_\sigma = 2$ ,  $c_h = 0.8$  and  $N_{tap} = 12$ . For this chosen model, we conducted training with identical model settings as outlined in 12, but with different values for coefficients  $c_h$ ,  $N_\sigma$  and  $N_{tap}$ .

In the first set of experiments, we set  $N_{tap} = 12$  and vary  $c_h$  and  $N_\sigma$ . Note that increasing the coefficient  $N_\sigma$  leads to a significant increase in computational time. We show different test errors in the Table 2. Once the coefficient  $c_h$  reaches a sufficiently high value (i.e.  $c_h \geq 0.8$ ), we observe no significant difference in test errors. Additionally, we do not find a high correlation between the error and the coefficient  $N_\sigma$ . We set the  $N_\sigma$  as low as possible, to a fixed value of  $N_\sigma = 2$ . Similarly, we set the coefficient  $c_h$  to a fixed value of 0.8.

In the second experiment, we fix  $N_\sigma = 2$  and  $c_h = 0.8$  and vary the number of taps  $N_{tap}$ . By increasing the number of taps, the computational time also increases. We show different test errors in the Table 3. Although there is an improvement of approximately 1.5% in the test error when  $N_{tap} = 20$  compared to when  $N_{tap} = 12$ , it comes at the cost of increased training time. Specifically, the training time per one epoch increases from  $4.37s$  to  $5.26s$ , representing more than 20% increase. Due to this significant increase in training time, but not very significant improvement in performance, we decide to fix the number of taps at  $N_{tap} = 12$ .

**Remark C.1.** *Given the above description, it is important to emphasize that, although in principle, the activation layer of CNO (2.3) needs to be exactly equivariant, i.e.,  $\sigma(\mathcal{B}_\omega) \subset \mathcal{B}_{\omega'}$  for the pair  $(\omega, \omega')$ , for the CNO architecture to be representation equivariant in the sense of [9], definition 3.4, see also section A.5, several approximations are used in practice that might be lead to this condition to hold only approximately. However, as the above results show, once the upsampling frequency is choosen high enough, this approximation of equivariance seems to suffice in practice, see also results in Section C.4. Nevertheless, if exact equivariance is sought for, it can be realized through either polynominal or rational activation functions as suggested in Remark B.4 although this choice might be of little practical utility.*

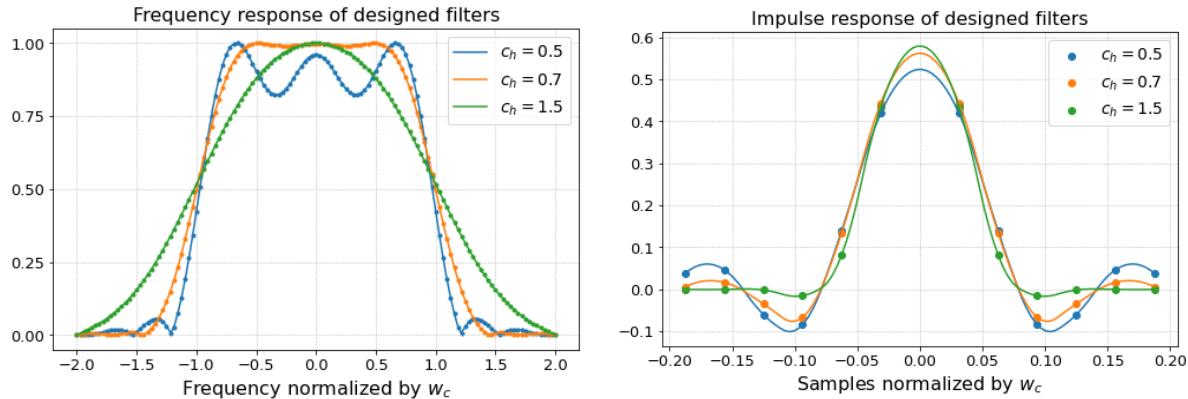


Figure 4: On the left: Frequency responses of different designed filters. On the right: Impulse responses of different designed filters. The sampling rate is  $s = 128$ , the cutoff frequency is  $w_c = s/2.0001$ , while the half-width of each filter is  $w_h = c_h \cdot s$ . Each filter has  $N_{tap} = 12$  taps.

In the simplest scenario, the architecture consists only of the lifting layer, number of (D) and (U) blocks, and the projection layer. In this simple scenario, once the input is lifted to higher dimensional space (in the channel width), one performs first  $T$  iterations of (D) blocks. These  $T$  iterations define the *encoder*, namely

$$v_{l+1} = \mathcal{D}_{s_l, s_{l+1}} \circ \Sigma_{s_l, s_{l+1}} \circ \mathcal{K}_{s_l}(v_l), \quad v_l \in \mathcal{B}_{s_l}(D, \mathbb{R}^{d_l}), \quad l = 0 \dots T-1,$$

where  $s_l = s/2^l$  is the current bandlimit and  $d_l$  is the current number of channels. The next  $T$  iterations are (U) blocks and are devoted to the *decoder*. Let  $\tilde{s}_l = s_{2T-l}$ . The decoder is defined as

$$v_{l+1} = \mathcal{U}_{\tilde{s}_l, \tilde{s}_{l+1}} \circ \Sigma_{\tilde{s}_l, \tilde{s}_{l+1}} \circ \mathcal{K}_{\tilde{s}_l}(v_l), \quad v_l \in \mathcal{B}_{\tilde{s}_l}(D, \mathbb{R}^{d_l}), \quad l = T \dots 2T-1.$$

Table 2: CNO model. Navier-Stokes Equations. Relative median  $L^1$ -error computed over 128 in-distribution testing samples for different filter designs. The error of the model with original filter parameters  $c_h = 0.8$ ,  $N_\sigma = 2$  and  $N_{tap} = 12$  is marked in blue.

	$c_h = 0.6$	$c_h = 0.8$	$c_h = 1.0$	$c_h = 1.5$	$c_h = 2.0$
$N_\sigma = 2$	2.87%	<b>2.76%</b>	2.77%	2.91%	2.86%
$N_\sigma = 3$	2.93%	2.86%	2.86%	2.87%	2.97%
$N_\sigma = 4$	2.80%	2.89%	2.88%	2.87%	2.89%
$N_\sigma = 5$	2.93%	2.84%	2.88%	2.98%	2.89%
$N_\sigma = 6$	3.02%	2.86%	2.88%	2.99%	2.82%

Table 3: CNO model. Navier-Stokes Equations. Relative median  $L^1$ -error computed over 128 in-distribution testing samples for different number of taps  $N_{tap}$ . The error of the model with original filter parameters  $c_h = 0.8$ ,  $N_\sigma = 2$  and  $N_{tap} = 12$  is marked in blue.

	$N_{tap} = 12$	$N_{tap} = 16$	$N_{tap} = 20$	$N_{tap} = 24$
$N_\sigma = 2 \& c_h = 0.8$	<b>2.76%</b>	2.72%	2.70%	2.75%

The last output of the decoder is projected to the output space (in the channel width). In all the experiments, we use  $d_l = d_e/2$  as the lifting dimension. In the encoder, the number of channels increases as per

$$d_e/2 \mapsto d_e \mapsto 2d_e \mapsto \dots \mapsto 2^{T-1}d_e.$$

The number  $d_e$  is a hyperparameter. In this simple case where *no* UNet style patching is present in the architecture, the number of channels in the decoder decreases as per

$$2^{T-1}d_e \mapsto 2^{T-2}d_e \mapsto \dots \mapsto d_e$$

When the patching is present in the architecture (see Figure 1), number of channels in the decoder changes differently (as a certain number of transferred channels is concatenated).

**Operator UNet architecture.** We add 2 (I) block 2.8 before each upsampling block. One block is applied before patching the additional channels, while the other is applied after patching. Additionally, we add a few (R) blocks 2.7 between each level of the encoder and decoder. We denote the number of residual blocks in the bottleneck of the network as a hyperparameter  $N_{res,b}$ , while the number of (R) blocks in the intermediate levels is denoted by  $N_{res,i}$  (each level has the same number of (R) blocks). Throughout our training and testing, we fix the size of the

convolution kernels to  $k = 3$ . Moreover, we apply *batch normalization* after each convolution operation, except in the lifting and the projection layers.

**Remark C.2.** *The objectives of cross-validation are  $T$ ,  $d_e$ ,  $N_{res,b}$  and  $N_{res,i}$ .*

### C.1.5 Galerkin Transformer

The Galerkin Transformer (abbreviated as GT) as presented in [8] is a model founded on attention-based operator learning. Central to its design is a "softmax-free" attention mechanism. Structurally, GT is an encoder-decoder model, and it uses the Galerkin-type transformer at its architectural bottleneck.

The encoder's role is to convert the input into the latent feature domain. Its design comprises 4 convolutional layers, which incrementally downscale the input's dimensions while enlarging its channel dimensions. Further enhancing its function, the encoder incorporates positional encoding at a coarser level, which is then combined with the extracted features and forwarded to the bottleneck.

The features are flattened, paving the way for the application of scaled dot-product multi-head attention. Let us characterize the single-head Galerkin-type attention: Given an input embedding  $y \in \mathbb{R}^{n \times d}$ , and using trainable matrices  $W_Q, Q_K, Q_V \in \mathbb{R}^{d \times d}$ , we can determine the query, key, and value as  $Q = yW_Q$ ,  $K = yW_K$ , and  $V = yW_V$ , respectively. The formal representation of the Galerkin-type single-head attention, denoted as  $\text{Attn} : \mathbb{R}^{n \times d} \rightarrow \mathbb{R}^{n \times d}$  is

$$\text{Attn}(y) = y + Q(\tilde{K}^T \tilde{V})/n + g(y + Q(\tilde{K}^T \tilde{V})/n),$$

where  $g$  is a 2-layer FFNN and  $\tilde{\cdot}$  is the layer normalization.

Lastly, the decoder is made up of a convolutional neural network that upsamples the output of the transformer to a desired dimension and several spectral convolutional layers. For an in-depth understanding of spectral convolutional layers, one can refer to [33].

Convolutional neural network in the encoder uses *relu* activation function, while the one in the decoder uses *silu* activation function. Spectral layers in the decoder use *silu* activation function.

The objectives of the cross-validation are:

- number of attention blocks :  $n$
- number of heads in the attetion:  $h$
- latent dimension in the attetion:  $d$
- number of decoder layers:  $L$
- latent dimension of the decoder:  $d_v$
- number of Fourier modes of the decoder:  $k_{max}$

### C.1.6 DeepONet

Let  $x := (x_1, \dots, x_m) \in D$  be a fixed set of *sensor points*. Given an input function  $u \in \mathcal{X}$ , we encode it by the point values  $\mathcal{E}(u) = (u(x_1), \dots, u(x_m)) \in \mathbb{R}^m$ . DeepONet is formulated in terms of two neural networks [39]: (1) a *branch-net*  $\beta$ , which maps the point values  $\mathcal{E}(u)$  to coefficients  $\beta(\mathcal{E}(u)) = (\beta_1(\mathcal{E}(u)), \dots, \beta_p(\mathcal{E}(u)))$ , resulting in a mapping

$$\beta : \mathbb{R}^m \rightarrow \mathbb{R}^p, \quad \mathcal{E}(\bar{u}) \mapsto (\beta_1(\mathcal{E}(\bar{u})), \dots, \beta_p(\mathcal{E}(\bar{u}))). \quad (\text{C.5})$$

and (2) a *trunk-net*  $\tau(y) = (\tau_1(y), \dots, \tau_p(y))$ , which is used to define a mapping

$$\tau : U \rightarrow \mathbb{R}^p, \quad y \mapsto (\tau_1(y), \dots, \tau_p(y)). \quad (\text{C.6})$$

While the branch net provides the coefficients, the trunk net provides the “basis” functions in an expansion of the output function of the form

$$\mathcal{G}(u)(y) = \sum_{k=1}^p \beta_k(\bar{u}) \tau_k(y), \quad \bar{u} \in \mathcal{X}, \quad y \in U, \quad (\text{C.7})$$

with  $\beta_k(\bar{u}) = \beta_k(\mathcal{E}(\bar{u}))$ . The resulting mapping  $\mathcal{G} : \mathcal{X} \rightarrow \mathcal{Y}$ ,  $u \mapsto \mathcal{G}$  is a *DeepONet*.

In the numerical experiments, for the trunk-net we use simple feed-forward neural networks. On the other hand the branch consists of a convolutional network of the following form:

$$\mathcal{G} : \mathcal{X} \rightarrow \mathcal{Y} : \quad \mathcal{G} = Q \circ Fl \circ R_{N_{res}} \circ \dots \circ R_1 \circ D_M \circ I_M \circ \dots \circ D_1 \circ I_1 \quad (\text{C.8})$$

where  $I$ ,  $D$  and  $R$  are the *invariant*, *downsampling* and *ResNet* blocks defined in 2, where the downsampling in  $D$  and  $\Sigma$  is instead performed by average pooling with kernel size 2. The parameter  $r$  in the residual block is set to 1. The output is then flattened through  $Fl$  and linearly transformed by  $Q : \mathbb{R}^n \rightarrow \mathbb{R}^p$ , with  $n$  being the number of units after flattening. The convolution is performed with a kernel of size 3 and stride 1, whereas the number of channels across the layers is

$$32 \mapsto 64 \mapsto 128 \mapsto \dots \mapsto 2^{M-1}32.$$

The activation function is chosen as Leaky ReLU. The number of layers  $L$  and units  $d$  of the trunk, the number of layers  $M$  and residual blocks  $N_{res}$  of the branch, and the number of bases  $p$ , are chosen through cross-validation.

### C.1.7 Fourier Neural Operator

A *Fourier neural operator* (FNO)  $\mathcal{G}$  [33] is a composition

$$\mathcal{G} : \mathcal{X} \rightarrow \mathcal{Y} : \quad \mathcal{G} = Q \circ \mathcal{L}_T \circ \dots \circ \mathcal{L}_1 \circ R. \quad (\text{C.9})$$

It has a “lifting operator”  $u(x) \mapsto R(u(x), x)$ , where  $R$  is represented by a linear function  $R : \mathbb{R}^{d_u} \rightarrow \mathbb{R}^{d_v}$  where  $d_u$  is the number of components of the input function and  $d_v$  is the

“lifting dimension”. The operator  $Q$  is a non-linear projection, instantiated by a shallow neural network with a single hidden layer, 128 neurons and *GeLU* activation function, such that  $v^{L+1}(x) \mapsto \mathcal{G}(u)(x) = Q(v^{L+1}(x))$ .

Each *hidden layer*  $\mathcal{L}_\ell : v^\ell(x) \mapsto v^{\ell+1}(x)$  is of the form

$$v^{\ell+1}(x) = \sigma \left( W_\ell \cdot v^\ell(x) + \left( K_\ell v^\ell \right) (x) \right),$$

with  $W_\ell \in \mathbb{R}^{d_v \times d_v}$  a trainable weight matrix (residual connection),  $\sigma$  an activation function, corresponding to GeLU, and the *non-local Fourier layer*,

$$K_\ell v^\ell = \mathcal{F}_N^{-1} \left( P_\ell(k) \cdot \mathcal{F}_N v^\ell(k) \right),$$

where  $\mathcal{F}_N v^\ell(k)$  denotes the (truncated)-Fourier coefficients of the discrete Fourier transform (DFT) of  $v^\ell(x)$ , computed based on the given  $s$  grid values in each direction. Here,  $P_\ell(k) \in \mathbb{C}^{d_v \times d_v}$  is a complex Fourier multiplication matrix indexed by  $k \in \mathbb{Z}^d$ , and  $\mathcal{F}_N^{-1}$  denotes the inverse DFT.

The lifting dimension  $d_v$ , the number of Fourier layers  $L$  and  $k_{max}$ , defined in 2, are objectives of cross-validation.

## C.2 Training Details

The training of the models, including the baselines (except GT), is performed with the ADAM optimizer, with a learning rate  $\eta$  for 1000 epochs and minimizing the  $L^1$ -loss function. We also use a step learning rate scheduler and reduce the learning rate of each parameter group by a factor  $\gamma$  every epoch. We train FFNN, UNet, and DeepONet in mini-batches of size 10 and FNO and CNO in batches of 32. A weight decay of magnitude  $w$  is used. All the parameters mentioned above ( $\eta, \gamma, w$ ) are chosen through cross-validation.

The GT models are trained with ADAM optimizer, minimizing the weighted  $L^2$ -loss function (see [8] implementation for clarification). Number of epochs is 1000. A learning rate scheduler is set according to a *OneCycleLR* policy ( $\text{max\_lr} = 5 \cdot 10^{-4}$ ,  $\text{div\_fac} = 10^4$ ,  $\text{pct\_start} = 0.3$ ). The selection of  $\text{max\_lr}$  relies on empirical observations.

At every epoch, the relative  $L^1$  error is computed on the validation set, and the set of trainable parameters resulting in the lowest error during the entire process is saved for testing. Early stopping is used to interrupt the training if the best validation error does not improve after 50 epochs.

The cross-validation is performed by running a random search over a chosen range of hyperparameters values and selecting the configuration, realizing the lowest relative  $L^1$  error on the validation set. Overall, 30 hyperparameters configurations are tested for the FFNN, UNet and DeepONet, 48 to 72 configurations for GT, 24 to 48 configurations for CNO and 36 to 72 configurations for FNO. The model size (minimum and maximum number of trainable parameters) covered in this search are reported in Table 5.

The results of the random search, i.e., the best-performing hyperparameter configurations for each model and each benchmark, are reported in tables 6, 10 and 7, 11 and 12.

**Different Initialization.** After selecting the models and computing the test median errors, we proceed to train the CNO, FNO, and UNet models again using the same settings but different initializations for the model parameters (by changing the random seeds). Each model is trained for each experiment a total of 10 times. We report the means and the standard deviations of the 10 different test median errors for each benchmark experiment in the Table 4. We observe from this table that CNO is very robust with respect to random initializations, with very low standard deviation to mean ratio for all the benchmarks in the **RPB** dataset.

Table 4: Means and standard deviations for the 10 relative median  $L^1$  test errors, for both in-distribution testing, for the CNO, FNO and U-Net models. The format is  $mean \pm std$ .

	CNO	FNO	UNet
<b>Poisson Equation</b>	$0.34 \pm 0.09\%$	$4.88 \pm 0.18\%$	$0.76 \pm 0.16\%$
<b>Wave Equation</b>	$0.63 \pm 0.06\%$	$1.08 \pm 0.07\%$	$1.67 \pm 0.12\%$
<b>Smooth Transport</b>	$0.27 \pm 0.04\%$	$0.34 \pm 0.03\%$	$0.79 \pm 0.21\%$
<b>Discontinuous Transport</b>	$1.06 \pm 0.04\%$	$1.18 \pm 0.03\%$	$1.40 \pm 0.09\%$
<b>Allen-Cahn</b>	$0.67 \pm 0.09\%$	$0.28 \pm 0.03\%$	$1.84 \pm 0.33\%$
<b>Navier-Stokes</b>	$2.91 \pm 0.08\%$	$3.68 \pm 0.10\%$	$3.48 \pm 0.07\%$
<b>Darcy Flow</b>	$0.42 \pm 0.02\%$	$0.90 \pm 0.08\%$	$0.65 \pm 0.10\%$
<b>Compressible Euler</b>	$0.35 \pm 0.01\%$	$0.45 \pm 0.01\%$	$0.39 \pm 0.01\%$

### C.3 Details about the description and numerical results in each benchmark

This section provides details about all the experiments that are a part of the **RPB** benchmarks of the main text.

#### C.3.1 Poisson Equation

In this experiment, we study Poisson equation 4.1 with the source term given by

$$f(x, y) = \frac{\pi}{K^2} \sum_{i,j}^K a_{ij} \cdot (i^2 + j^2)^r \sin(\pi i x) \sin(\pi j y), \quad \forall (x, y) \in D,$$

where  $K = 16$ ,  $r = 0.5$  and  $a_{ij}$  are i.i.d. uniformly distributed from  $[-1, 1]$ . Given the source term above, the *exact* solution  $u$  of the Poisson equation is given by

$$u(x, y) = \frac{1}{\pi K^2} \sum_{i,j}^K a_{ij} \cdot (i^2 + j^2)^{r-1} \sin(\pi i x) \sin(\pi j y), \quad \forall (x, y) \in D.$$

Table 5: Minimum (Top sub-row) and maximum (Bottom sub-row) number of trainable parameters among the random-search hyperparameters configurations for all the models in every problem reported in Table 1 in main text.

	<b>FFNN</b>	<b>GT</b>	<b>ResNet</b>	<b>UNet</b>	<b>DON</b>	<b>FNO</b>	<b>CNO</b>
<b>Poisson Equation</b>	0.3M	8.5M	0.1M	0.5M	0.8M	0.2M	0.5M
	8.2M	19.1M	10.2M	31.0M	48.1M	18.9M	26.8M
<b>Wave Equation</b>	0.3M	8.5M	0.1M	0.5M	0.8M	0.2M	1.5M
	6.0M	19.1M	10.2M	31.0M	48.1M	7.9M	23.6M
<b>Smooth Transport</b>	0.3M	8.5M	0.1M	0.5M	0.7M	0.2M	0.5M
	8.2M	19.1M	10.2M	7.7M	49.2M	23.6M	18.9M
<b>Discontinuous Transport</b>	0.3M	8.5M	0.1M	0.5M	0.7M	0.2M	0.5M
	5.5M	19.1M	10.2M	7.7M	49.2M	23.6M	18.9M
<b>Allen-Cahn</b>	0.3M	2.1M	0.1M	0.5M	1.0M	0.9M	0.5M
	7.1M	19.1M	10.2M	31.0M	47.9M	65.6M	8.4M
<b>Navier-Stokes</b>	0.3M	2.1M	0.1M	0.5M	1.0M	0.2M	0.5M
	7.1M	19.1M	10.2M	31.0M	47.9M	65.6M	14.1M
<b>Darcy Flow</b>	0.3M	2.1M	0.1M	0.5M	1.0M	0.2M	0.5M
	7.1M	19.1M	10.2M	31.0M	47.9M	23.6M	8.4M
<b>Compressible Euler</b>	1.1M	2.1M	0.1M	0.5M	0.8M	0.2M	1.5M
	18.6M	19.1M	10.2M	31.0M	49.4M	23.6M	31.7M

Table 6: FFNN best-performing hyperparameters configuration for different benchmark problems.

	$\eta$	$\gamma$	$w$	$L$	$d$	Trainable Params
<b>Poisson Equation</b>	0.0005	0.98	1e-06	10	512	6.6M
<b>Wave Equation</b>	0.001	0.98	1e-06	4	256	2.3M
<b>Continuous Translation</b>	0.001	1.0	0.0	16	256	3.1M
<b>Discontinuous Translation</b>	0.0005	1.0	0.0	6	512	5.5M
<b>Allen-Cahn</b>	0.0005	0.98	0.0	8	512	6.0M
<b>Navier-Stokes</b>	0.001	1.0	1e-06	16	256	3.1M
<b>Darcy Flow</b>	0.0005	0.98	1e-06	16	256	3.1M
<b>Compressible Euler</b>	0.0005	1.0	0.0	16	32	1.1M

During the out-of-distribution testing, we augment the number of modes to  $K = 20$  and evaluate the models' ability to generalize to inputs with frequencies higher than those encountered during training. We approximate the operator  $\mathcal{G}^\dagger$ , which maps  $f$  to  $u$ . An illustration of the

Table 7: UNet best-performing hyperparameters configuration for different benchmark problems.

	$\eta$	$\gamma$	$w$	$c$	Trainable Params
<b>Poisson Equation</b>	0.001	0.98	0.0	32	7.8M
<b>Wave Equation</b>	0.001	1.0	1e-06	64	31.0M
<b>Continuous Translation</b>	0.001	0.98	1e-06	16	1.9M
<b>Discontinuous Translation</b>	0.001	0.98	1e-06	32	7.8M
<b>Allen-Cahn</b>	0.0005	0.98	1e-06	64	31.0M
<b>Navier-Stokes</b>	0.0005	0.98	1e-06	64	31.0M
<b>Darcy Flow</b>	0.001	0.98	0.0	32	7.8M
<b>Compressible Euler</b>	0.001	0.98	1e-06	32	7.8M

Table 8: ResNet best-performing hyperparameters configuration for different benchmark problems.

	$\eta$	$\gamma$	$w$	$c$	$N_{res}$	Trainable Params
<b>Poisson Equation</b>	0.001	0.98	0.0	8	8	0.2M
<b>Wave Equation</b>	0.001	0.98	1e-06	16	8	0.6M
<b>Continuous Translation</b>	0.001	0.98	1e-06	32	6	2.0M
<b>Discontinuous Translation</b>	0.001	0.98	1e-06	32	4	1.4M
<b>Allen-Cahn</b>	0.001	0.98	1e-06	32	4	1.4M
<b>Navier-Stokes</b>	0.001	0.98	1e-06	64	8	10.2M
<b>Darcy Flow</b>	0.001	0.98	0.0	8	8	0.2M
<b>Compressible Euler</b>	0.001	1.0	0.0	32	8	2.6M

operator  $\mathcal{G}^\dagger$  is given in the Figure 28. For training purposes, we generate 1024 samples and for testing, we generate 256 samples for both in-distribution and out-of-distribution testing, by sampling the exact solution  $u$  at a resolution of  $64 \times 64$  points on  $D = [0, 1]^2$ . We also create a validation set consisting of 128 samples for model selection. The training data is normalized to the interval  $[0, 1]$ . The testing data is normalized with the same normalization constants as the training data. In Figure 5, we show empirical test error distributions for UNet, FNO and CNO models (in-distribution in the left Figure and out-of-distribution in the right Figure). We show a random in-distribution testing sample and an out-of-distribution testing sample, as well as

Table 9: DeepONet best-performing hyperparameters configuration for different benchmark problems.

	$\eta$	$\gamma$	$w$	$p$	$L$	$d$	$M$	$N_{res}$	Trainable Params
<b>Poisson Equation</b>	0.001	0.98	0.0	500	8	128	4	4	5.2M
<b>Wave Equation</b>	0.0005	0.98	0.0	100	4	512	4	4	4.2M
<b>Continuous Translation</b>	0.0005	0.98	0.0	500	8	128	4	0	2.8M
<b>Discontinuous Translation</b>	0.0005	0.98	0.0	100	8	512	4	4	5.3M
<b>Allen Cahn</b>	0.0005	0.98	1e-06	50	8	512	4	4	5.0M
<b>Navier Stokes</b>	0.0005	0.98	1e-06	100	8	512	6	2	30.3M
<b>Darcy Flow</b>	0.0005	0.98	0.0	500	8	512	4	4	7.1M
<b>Compressible Euler</b>	0.0005	0.98	1e-06	500	8	256	4	4	11.7M

Table 10: Galerkin Transformer best-performing hyperparameters configuration for different benchmark problems.

	$n$	$h$	$d$	$L$	$d_v$	$k_{max}$	Trainable Params
<b>Poisson Equation</b>	4	4	64	2	64	16	8.6M
<b>Wave Equation</b>	4	4	64	2	64	16	8.6M
<b>Continuous Translation</b>	4	2	128	3	64	16	17.4M
<b>Discontinuous Translation</b>	8	2	128	4	64	16	17.4M
<b>Allen-Cahn</b>	2	4	128	2	32	16	2.5M
<b>Navier-Stokes</b>	2	2	256	2	64	16	10.0M
<b>Darcy Flow</b>	4	2	256	2	32	16	4.4M
<b>Compressible Euler</b>	2	4	64	4	64	16	16.9M

predictions made by CNO, FNO and UNet in Figure 6. As was already evidenced in Table 1 of the main text, Figure 5 demonstrates that CNO is clearly the best performing model here with U-Net a distant second. FNO performs very poorly on this problem, with test errors that are more than an order of magnitude higher than CNO. A closer perusal of Figure 6 reveals that FNO approximates the multiple scales in the exact solution very poorly and this is particularly striking for the out of distribution testing example shown in this figure. On the other hand, CNO approximates the multiple frequencies in the solution very accurately.

Table 11: FNO best-performing hyperparameters configuration for different benchmark problems.

	$\eta$	$\gamma$	$w$	$pad$	$k_{max}$	$d_v$	$L$	Trainable Params
<b>Poisson Equation</b>	0.001	0.98	1e-6	0	16	16	5	0.7M
<b>Wave Equation</b>	0.001	0.98	1e-6	0	20	16	4	0.8M
<b>Smooth Transport</b>	0.001	0.98	1e-6	4	20	32	5	4.1M
<b>Discontinuous Transport</b>	0.001	0.98	1e-6	4	16	32	5	2.6M
<b>Allen-Cahn</b>	0.001	0.98	1e-6	0	20	16	3	0.6M
<b>Navier-Stokes</b>	0.001	0.98	1e-6	0	16	128	5	42.1M
<b>Darcy Flow</b>	0.001	0.98	1e-6	0	24	16	2	0.6M
<b>Compressible Euler</b>	0.001	0.98	1e-6	8	24	32	3	3.6M

Table 12: CNO best-performing hyperparameters configuration for different benchmark problems.

	$\eta$	$\gamma$	$w$	$M$	$d_e$	$N_{res,b}$	$N_{res,i}$	Trainable Params
<b>Poisson Equation</b>	0.001	0.98	1e-6	3	16	6	4	0.7M
<b>Wave Equation</b>	0.001	0.98	1e-10	3	48	6	4	6.6M
<b>Smooth Transport</b>	0.001	0.98	1e-6	3	32	6	2	2.8M
<b>Discontinuous Transport</b>	0.001	0.98	1e-6	3	32	4	5	2.5M
<b>Allen-Cahn</b>	0.001	0.98	1e-6	3	48	8	4	3.5M
<b>Navier-Stokes</b>	0.001	0.98	1e-10	3	32	8	1	3.3M
<b>Darcy Flow</b>	0.001	0.98	1e-6	3	48	4	4	5.3M
<b>Compressible Euler</b>	0.001	0.98	1e-10	4	48	8	1	7.3M

Finally, to further investigate the poor performance of FNO, as compared to CNO, for this problem, we present in Figure 7, the averaged logarithmic amplitude spectra, which compare the ground truth, CNO, FNO, and UNet models. We see from this spectrogram that i) the ground truth solution contains multiple scales, corresponding to a range of frequencies ii) the CNO model successfully captures the complete spectra with high accuracy, iii) FNO (and to some extent UNet) resolves the underlying spectrum with quite a lot of error, particularly in the high-frequency components, perhaps attributable to aliasing errors in this case.

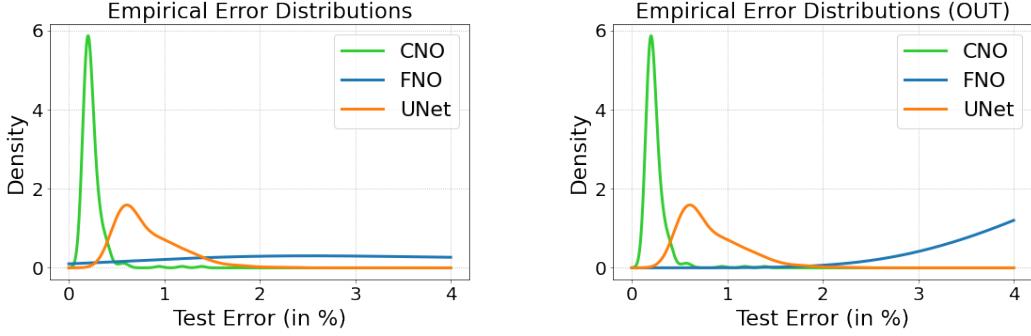


Figure 5: Poisson equation. Empirical test error distributions for UNet, FNO and CNO. Left: In-distribution testing. Right: Out-of-distribution testing.

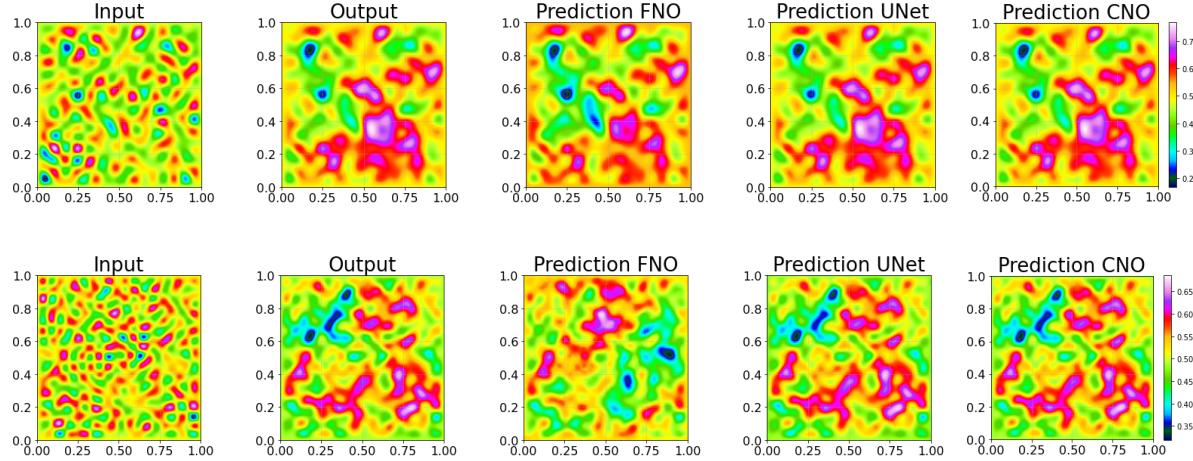


Figure 6: Poisson equation. Exact and predicted coefficients for an in-distribution (top row) and an out-of-distribution (bottom row) samples and for different models (columns). From left to right: input, output (ground truth), FNO, UNet and CNO.

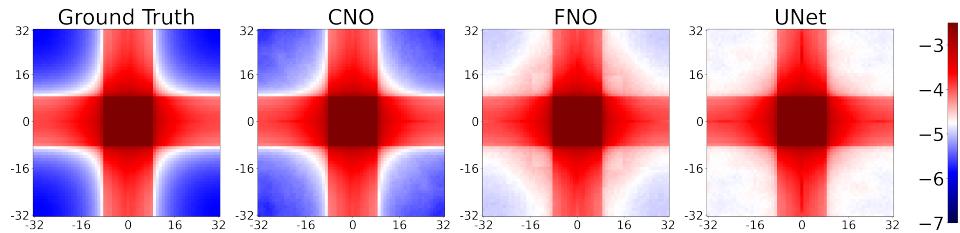


Figure 7: Poisson equation. Averaged logarithmic amplitude spectra comparing Ground Truth, CNO, FNO and UNet.

### C.3.2 Wave Equation

In this experiment, we study Wave equation 4.3 with constant speed of propagation  $c = 0.1$  and the initial condition given by 4.2 with  $K = 24$  and  $r = 1$ . The exact solution at time  $t > 0$  is given by

$$u(x, y, t) = \frac{\pi}{K^2} \sum_{i,j}^K a_{ij} \cdot (i^2 + j^2)^{-r} \sin(\pi i x) \sin(\pi j y) \cos\left(c\pi t \sqrt{i^2 + j^2}\right), \quad \forall (x, y) \in D.$$

The objective is to approximate the operator  $\mathcal{G}^\dagger : f \mapsto u(\cdot, T = 5)$ . An illustration of  $\mathcal{G}^\dagger$  is given in Figure 29. During the out-of-distribution testing, we decrease the decay parameter to  $r = 0.85$ . This adjustment changes the ratio between the amplitudes of different modes, which alters the dynamics of the solution. For the training set, we generate a total of 512 samples. In addition, we generate 256 samples for both in-distribution and out-of-distribution testing, all by sampling the above exact solution at a resolution of  $64 \times 64$ . Furthermore, we create a validation set comprising 128 samples. The training data is normalized to the interval  $[0, 1]$ . The testing data is normalized with the same normalization constants as the training data.

In Figure 8, we present the empirical test error distributions for UNet, FNO and CNO models during in-distribution and out-of-distribution testing. We also show a random in-distribution testing sample and a random out-of-distribution testing sample, as well as predictions made by CNO, FNO and UNet in Figure 9. Both these figures demonstrate that CNO is the best performing model in this case, reinforcing the conclusion of Table 1 of the main text.

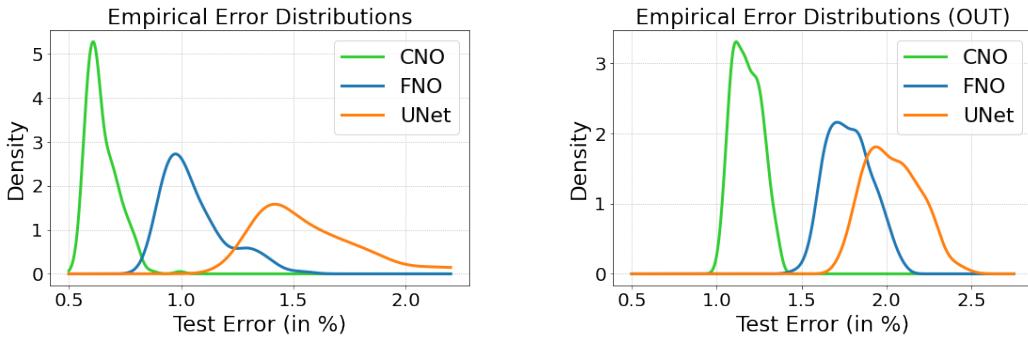


Figure 8: Wave equation. Empirical test error distributions for UNet, FNO and CNO. Left: In-distribution testing. Right: Out-of-distribution testing.

### C.3.3 Transport Equation

In this experiment, we study Transport equation 4.4. We fix the velocity field to  $v = (v_x, v_y) = (0.2, -0.2)$  leading to solution  $u(x, y, t) = f(x - v_x t, y - v_y t)$ . We conduct two different experiments, i.e., Smooth Transport and Discontinuous Transport. In both cases, the goal is to approximate the operator  $\mathcal{G}^\dagger : f \mapsto u(\cdot, T = 1)$ . Moreover, in both cases, we generate 512 training samples, 256 validation samples and 256 in-distribution and out-of-distribution testing samples, all from the exact solution. Each sample is normalized to the interval  $[0, 1]$ .

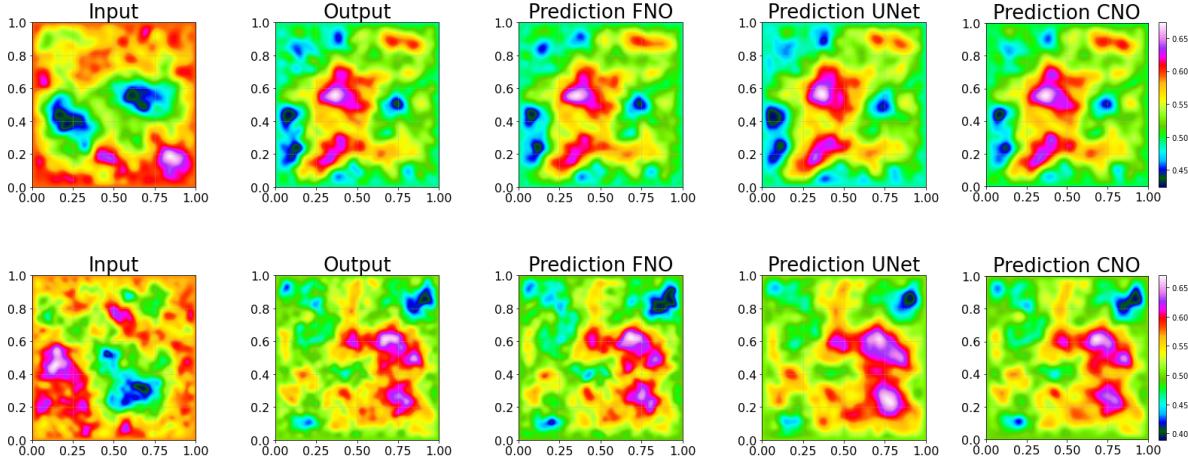


Figure 9: Wave equation. Exact and predicted coefficients for an in-distribution (top row) and an out-of-distribution (bottom row) samples and for different models (columns). From left to right: input, output (ground truth), CNO, FNO and UNet.

**Smooth Transport.** In this case, the data takes form of a radially symmetric Gaussian. The data is drawn from a Gaussian distribution with centers randomly and uniformly drawn from  $(0.2, 0.4)^2$  and corresponding variance drawn uniformly from  $(0.003, 0.009)$ . Formally, the initial conditions are given by

$$f(\underline{x}) = \frac{1}{\sqrt{(2\pi)^2 \det(\Sigma)}} \exp\left(-\frac{1}{2}(\underline{x} - \mu)^T \Sigma^{-1}(\underline{x} - \mu)\right), \quad \underline{x} = (x, y), \quad \mu = (\mu_x, \mu_y),$$

where  $\Sigma = \sigma I$  such that  $\sigma \sim \mathcal{U}(0.003, 0.009)$  and  $\mu_x, \mu_y \sim \mathcal{U}(0.2, 0.4)$ . Here,  $I$  is the identity matrix and  $\mathcal{U}(\cdot)$  is the uniform distribution. Finally, each initial condition is normalized to  $(0, 1)$ .

For *out-of-distribution* testing, the centers of the Gaussian inputs are sampled uniformly from  $(0.4, 0.6)^2$  (i.e.  $\mu_x, \mu_y \sim \mathcal{U}(0.4, 0.6)$ ). The data is generated at  $64 \times 64$  resolution. An illustration of the operator  $\mathcal{G}^\dagger$  for the Smooth Transport experiment is shown in Figure 30. We show empirical test error distributions for UNet, FNO and CNO models (in-distribution and out-of-distribution testing) in Figure 10. We show a random in-distribution testing sample and an out-of-distribution testing sample, as well as predictions made by CNO, FNO and UNet in Figure 11. The figures reinforce the conclusions drawn from Table 1 i.e., CNO is slightly superior to UNet and FNO for in-distribution testing. However, there is a significant advantage for CNO over UNet on out-of-distribution testing. On the other hand, FNO generalizes poorly out-of-distribution, as clearly seen from the sample shown in Figure 11. Similarly, DeepONet and FFNN are even poorer in terms of their generalization abilities, justifying the very high errors seen in Table 1. An example of this very poor generalization for DeepONet and FFNN can be seen in Figure 12.

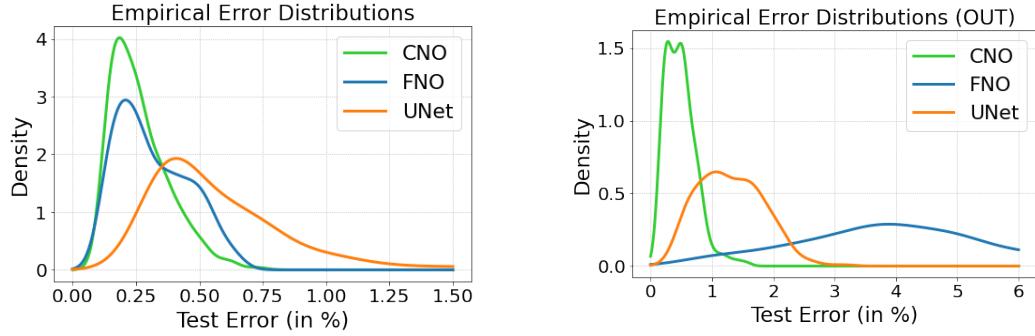


Figure 10: Smooth Transport. Empirical test error distributions for UNet, FNO and CNO. Left: In-distribution testing. Right: Out-of-distribution testing.

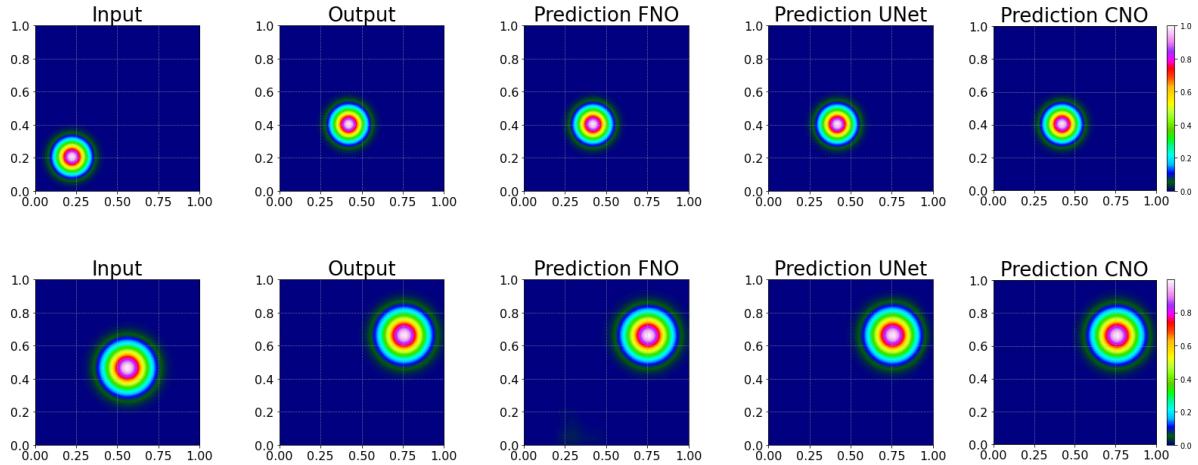


Figure 11: Smooth Transport. Exact and predicted coefficients for an in-distribution (top row) and an out-of-distribution (bottom row) samples and for different models (columns). From left to right: input, output (ground truth), CNO, FNO and UNet.

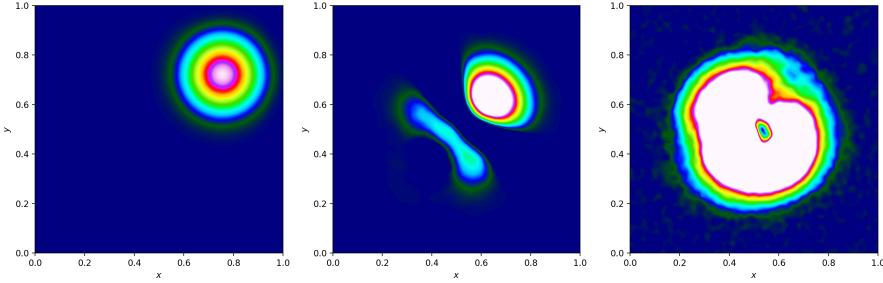


Figure 12: Smooth Transport. An out-of-distribution sample and predictions for DeepONet and FFNN. From left to right: output (ground truth), DeepONet and FFNN.

**Discontinuous Transport.** In this case, initial data in the form of the indicator function of radial disk with centers, uniformly drawn from  $(0.2, 0.4)^2$  and radii uniformly drawn from  $(0.1, 0.2)$ . For *out-of-distribution* testing, the centers of the disk are drawn uniformly from  $(0.4, 0.6)^2$ . Formally, the initial conditions are given by

$$f(\underline{x}) = \mathbb{1}_{S_r(\mu)}(\underline{x}), \quad \underline{x} = (x, y), \quad \mu = (\mu_x, \mu_y),$$

where  $r \sim \mathcal{U}(0.1, 0.2)$  and  $\mu_x, \mu_y \sim \mathcal{U}(0.2, 0.4)$ . Also,  $\mathbb{1}_\cdot$  is an indicator function and  $S_r(\mu)$  is the sphere of radius  $r$  with the center  $\mu$ , defined by

$$S_r(\mu) = \{\underline{x} : \|\underline{x} - \mu\|_2 \leq r\}.$$

Note that discontinuous data has infinite spectral content, so the aliasing error is always present when the data is sampled. For that reason, we first generate the samples at  $128 \times 128$  resolution, to reduce the aliasing error that emerges in data generation. We get our actual samples by downsampling the generated data in the frequency domain to the resolution  $64 \times 64$ . As the Gibbs phenomenon is strongly present when discontinuous data is downsampled in this way, we reduce the impact of this phenomenon by applying a Gaussian filter with a standard deviation  $\sigma = 1.75$  to the generated samples, before downsampling them to the final resolution. An example of a random sample with horizontal cut plots is shown in the Figure 13.

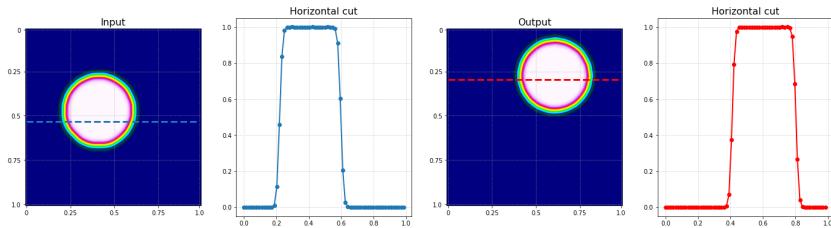


Figure 13: Discontinuous Transport. An example with horizontal cut plots of the disks.

We also plot empirical test error distributions for UNet, CNO and FNO models (in-distribution and out-of-distribution testing) in Figure 14. We plot a random in-distribution testing sample and an out-of-distribution testing sample, as well as predictions made by CNO, FNO and UNet

in Figure 15. These figures clearly reinforce the conclusions from Table 1 that UNet performs as good as the CNO on out-of-distribution testing. On the other hand, FNO, DeepONet, FFNN and GT (in that order) generalize very poorly as they fail to be translation equivariant.

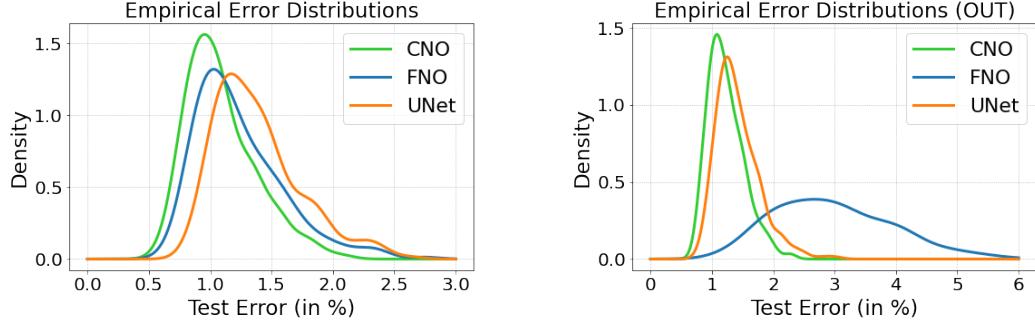


Figure 14: Discontinuous Transport. Empirical test error distributions for UNet, FNO and CNO. Left: In-distribution testing. Right: Out-of-distribution testing.

### C.3.4 Allen-Cahn Equation

In this experiment, we study Allen-Cahn equation 4.5 with fixed reaction rate  $\varepsilon = 220$  and initial condition given by 4.2 with  $K = 24$  and  $r = 1$ . The goal is to approximate the operator  $\mathcal{G}^\dagger : f \mapsto u(\cdot, T = 0.0002)$  (see Figure 32 for illustrations).

As exact solutions are no longer available, we generate the training and test data using a standard finite difference discretization of the Allen-Cahn equation. We uniformly discretize space at the resolution  $s^2 = 64 \times 64$  and set  $\Delta x = 1/s$ . As we are using an explicit method, we uniformly discretize the time domain with the time step  $\Delta t \approx 5.47 \cdot 10^{-7}$  and set  $N = \lfloor T/\Delta t \rfloor + 1$ . We denote  $U_{i,j}^n = u(i\Delta x, j\Delta x, n\Delta t)$  for  $i, j = 0, 1, \dots, s$  and  $n = 0, 1, \dots, N$ . Additionally, we also add the zero-valued ghost cells at the boundaries. The Finite Difference scheme is given by

$$U_{i,j}^{n+1} = U_{i,j}^n + \frac{\Delta t}{\Delta x} (U_{i+1,j}^n + U_{i,j+1}^n + U_{i-1,j}^n U_{i,j-1}^n - 4U_{i,j}^n) - \Delta t \varepsilon^2 U_{i,j}^n (U_{i,j}^n \cdot U_{i,j}^n - 1),$$

for  $i, j = 0, 1, \dots, s$  and  $n = 0, 1, \dots, N$ . With our choice of  $\Delta t$ , the CFL condition  $\Delta t < \frac{(\Delta x)^2}{2\varepsilon}$  is satisfied. We generate 256 training samples, 128 validation samples and 128 in-distribution and out-of-distribution testing samples, all at the  $64 \times 64$  resolution. The training data is normalized to the interval  $[0, 1]$ . The testing data is normalized with the same normalization constants as the training data. In Figure 16, we present the empirical test error distributions for UNet, FNO and CNO models during in-distribution and out-of-distribution testing. We also plot a random in-distribution testing sample and an out-of-distribution testing sample, as well as predictions made by CNO, FNO and UNet in Figure 17. Again, these figures reinforce the conclusions of Table 1 as FNO is marginally superior to CNO and UNet on in-distribution testing whereas UNet is the best model on out-of-distribution testing.

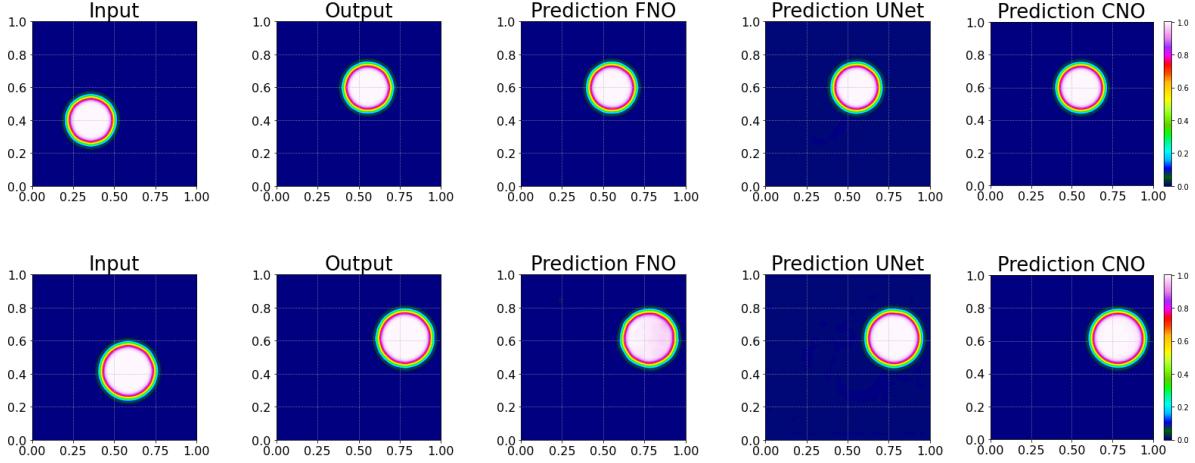


Figure 15: Discontinuous Transport. Exact and predicted coefficients for an in-distribution (top row) and an out-of-distribution (bottom row) samples and for different models (columns). From left to right: input, output (ground truth), CNO, FNO and UNet.

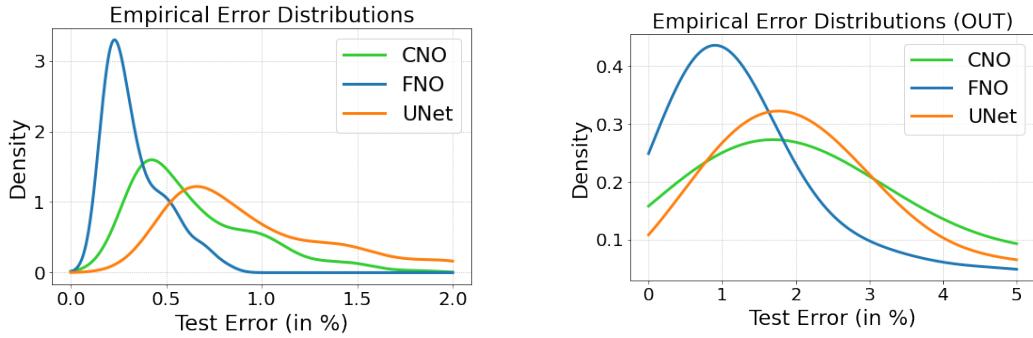


Figure 16: Allen-Cahn equation. Empirical test error distributions for UNet, FNO and CNO. Left: In-distribution testing. Right: Out-of-distribution testing.

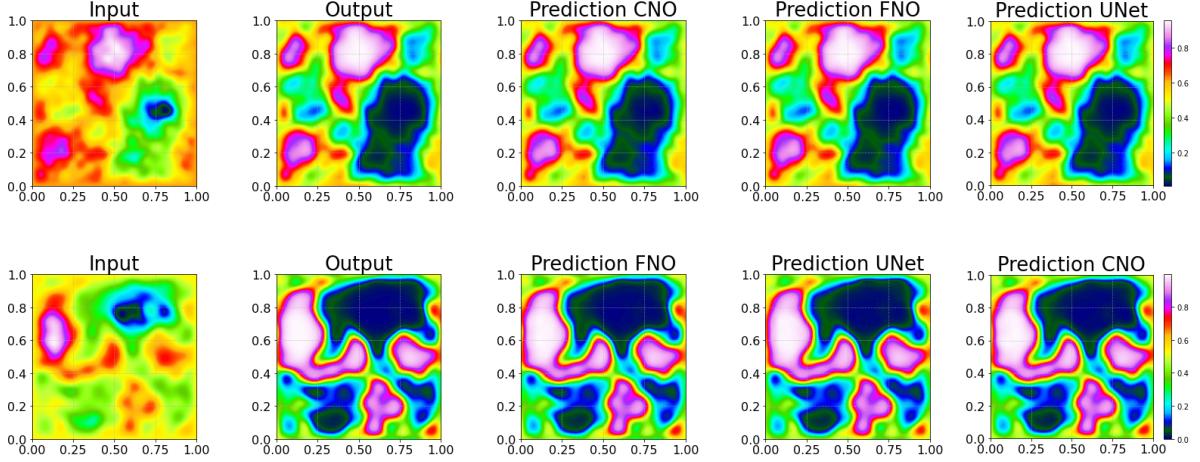


Figure 17: Allen-Cahn equation. Exact and predicted coefficients for an in-distribution (top row) and an out-of-distribution (bottom row) samples and for different models (columns). From left to right: input, output (ground truth), CNO, FNO and UNet.

### C.3.5 Navier-Stokes

In this experiment, we study a motion of an incompressible fluid with high Reynolds number. We study Navier-Stokes equations 4.6 in the torus  $D = \mathbb{T}^2$  with periodic boundary conditions and, for stabilization, viscosity  $\nu = 4 \times 10^{-4}$  only applied to high-enough Fourier modes. We take as initial conditions

$$u_0(x, y) = \begin{cases} \tanh\left(2\pi \frac{y-0.25}{\rho}\right) & \text{for } y + \sigma_\delta(x) \leq \frac{1}{2} \\ \tanh\left(2\pi \frac{0.75-y}{\rho}\right) & \text{otherwise} \end{cases} \quad (C.10)$$

$$v_0(x, y) = 0$$

where  $\sigma_\delta : [0, 1] \rightarrow \mathbb{R}$  is a perturbation of the initial data given by

$$\sigma_\delta(x) = \delta \sum_{k=1}^p \alpha_k \sin(2\pi kx - \beta_k). \quad (C.11)$$

The random variables  $\alpha_k$  and  $\beta_k$  are i.i.d. uniformly distributed on  $[0, 1]$  and  $[0, 2\pi]$  respectively. The parameters  $\delta$  and  $p$  are chosen to be  $\delta = 0.025$  and  $p = 10$ . For the smoothing parameter we choose  $\rho = 0.1$ . (see Figure 33 for illustrations). For the out-of-distribution experiments, we reduced  $\rho$  to  $\rho = 0.09$  and shifted the location of the shear layers towards the middle of the domain so that they were located at  $y = 0.3$  and  $y = 0.7$  instead of  $y = 0.25$  and  $y = 0.75$  like in the original initial condition.

Fix a mesh width  $\Delta = \frac{1}{N}$  for some  $N \in \mathbb{N}$ . We consider the following discretization of the

Navier-Stokes equations 4.6 in the Fourier domain

$$\begin{cases} \partial_t u^\Delta + \mathcal{P}_N(u^\Delta \cdot \nabla u^\Delta) + \nabla p^\Delta = \varepsilon_N |\nabla|^{2s}(Q_N * u^\Delta) \\ \nabla \cdot u^\Delta = 0 \\ u^\Delta|_{t=0} = \mathcal{P}_N u_0 \end{cases} \quad (\text{C.12})$$

where  $\mathcal{P}_N$  is the spatial Fourier projection operator mapping a function  $f(x, t)$  to its first  $N$  Fourier modes:  $\mathcal{P}_N = \sum_{|k|_\infty \leq N} \hat{f}_k(t) e^{ik \cdot x}$ . We additionally have the hyperviscosity parameter  $s \geq 1$  which can be used to dampen the higher Fourier modes strongly, thus allowing for a larger part of the spectrum to be free of numerical dissipation. The artificial viscosity term we use for the stabilization of the solver consists of a resolution-dependent viscosity  $\varepsilon_N$  and a Fourier multiplier  $Q_N$  controlling the strength at which different Fourier modes are damped. This allows us to not dampen the low frequency modes, while applying some diffusion to the problematic higher frequencies. The Fourier multiplier  $Q_N$  is of the form

$$Q_N(\mathbf{x}) = \sum_{\mathbf{k} \in \mathbb{Z}^d, |\mathbf{k}| \leq N} \hat{Q}_{\mathbf{k}} e^{i\mathbf{k} \cdot \mathbf{x}}. \quad (\text{C.13})$$

In order to have convergence, the Fourier coefficients of  $Q_N$  need to fulfill [27], [53] [54]

$$\hat{Q}_k = 0 \text{ for } |k| \leq m_N, 1 - \left( \frac{m_N}{|k|} \right)^{\frac{2s-1}{\theta}} \leq \hat{Q}_k \leq 1 \quad (\text{C.14})$$

where we have introduced an additional parameter  $\theta > 0$ . The quantities  $m_N$  and  $\varepsilon_N$  are required to scale as

$$m_N \sim N^\theta, \varepsilon_N \sim \frac{1}{N^{2s-1}}, 0 < \theta < \frac{2s-1}{2s}. \quad (\text{C.15})$$

For the experiment described here, we choose  $s = 1$ ,  $m_N = \sqrt{N}$ ,  $\varepsilon_N = \frac{0.05}{N}$ , and  $N = 128$ . This gives rise to the viscosity  $\nu \approx 4 \cdot 10^{-4}$  mentioned above.

Applying the Fourier projection operator to the PDE C.12 causes the solutions to be bandlimited functions and therefore they only have finitely many nonzero basis function coefficients (at most  $N$ ). By writing the above discretization in the Fourier basis, we transform the spatial derivatives into multiplications with the wave vectors  $k$  and obtain

$$\partial_t \hat{u}_k + ik^T \cdot \hat{B}_k + ik \hat{p}_k = -\nu |k|^2 \hat{u}_k \quad (\text{C.16})$$

where we have substituted  $B = u \otimes u$ . By requiring  $\hat{u}_k$  (and  $\partial_t \hat{u}_k$ ) to be divergence free, we can compute the pressure  $\hat{p}_k$  to be

$$\hat{p}_k = -\frac{k^T \cdot \hat{B}_k \cdot k}{|k|^2}. \quad (\text{C.17})$$

Note that the pressure can be computed from local quantities only. This is in contrast to numerical methods solving the equations in physical space where the pressure is obtained as the solution to a Poisson equation. Finally, we can solve the incompressible Euler equations by computing

$$\partial_t \hat{u}_k + \left( \text{Id} - \frac{kk^T}{|k|^2} \right) \cdot \hat{b}_k = -\nu |k|^2 \hat{u}_k \quad (\text{C.18})$$

where  $\hat{b}_k = ik^T \cdot \hat{B}_k$ . Timestepping is done using a third-order strong stability preserving Runge-Kutta scheme (SSPRK3)

$$\begin{aligned} u^{(1)} &= u(t) + \Delta t \partial_t u(t) \\ u^{(2)} &= \frac{3}{4}u(t) + \frac{1}{4}u^{(1)} + \frac{1}{4}\Delta t \partial_t u^{(1)} \\ u(t + \Delta t) &= \frac{1}{3}u(t) + \frac{2}{3}u^{(2)} + \frac{2}{3}\Delta t \partial_t u^{(2)}. \end{aligned} \quad (\text{C.19})$$

Note that through the construction of the pressure field, the numerical scheme is not exactly divergence-free. It merely preserves the divergence of the initial conditions  $u_0$ . We therefore implicitly project all the initial conditions onto divergence free vector fields. This operation is described by the Leray projection  $\mathbb{P} : L^2(\Omega) \rightarrow \{u \in L^2(\Omega) \mid \operatorname{div} u = 0\}$  mapping  $u \mapsto u - \nabla \Delta^{-1}(\operatorname{div} u)$ . In Fourier space, this can again be simplified to the local equation

$$\mathbb{P}\hat{u}_k = \left( \operatorname{Id} - \frac{kk^T}{|k|^2} \right) \cdot \hat{u}_k. \quad (\text{C.20})$$

For the training set, we generate a total of 750 samples. In addition, we generate 128 samples for validation set, in-distribution and out-of-distribution testing. To generate the training and test data, we simulate the Navier-Stokes equations with a spectral viscosity method on a  $128 \times 128$  resolution and downsample the data to a  $64 \times 64$  resolution. The goal is to learn the operator mapping the initial velocity to velocity at  $T = 1$ . The training data is normalized to the interval  $[0, 1]$ . The testing data is normalized with the same normalization constants as the training data. In Figure 18, we present the empirical test error distributions for UNet, FNO and CNO models during in-distribution and out-of-distribution testing. We also plot a random in-distribution testing sample and an out-of-distribution testing sample, as well as predictions made by CNO, FNO and UNet in Figure 19. These figures demonstrate that CNO is clearly the best performing model, for both in-distribution and out-of-distribution testing, outperforming UNet and FNO significantly. Moreover, given the highly multiscale nature of this problem (see Figure 2 of Main Text for spectrograms), it is not surprising that the errors with all the models are higher than in the other **RPB** benchmarks.

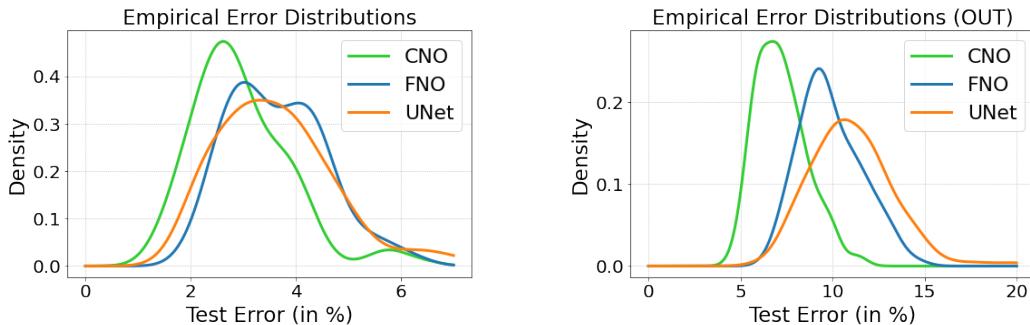


Figure 18: Navier-Stokes equations. Empirical test error distributions for UNet, FNO and CNO. Left: In-distribution testing. Right: Out-of-distribution testing.

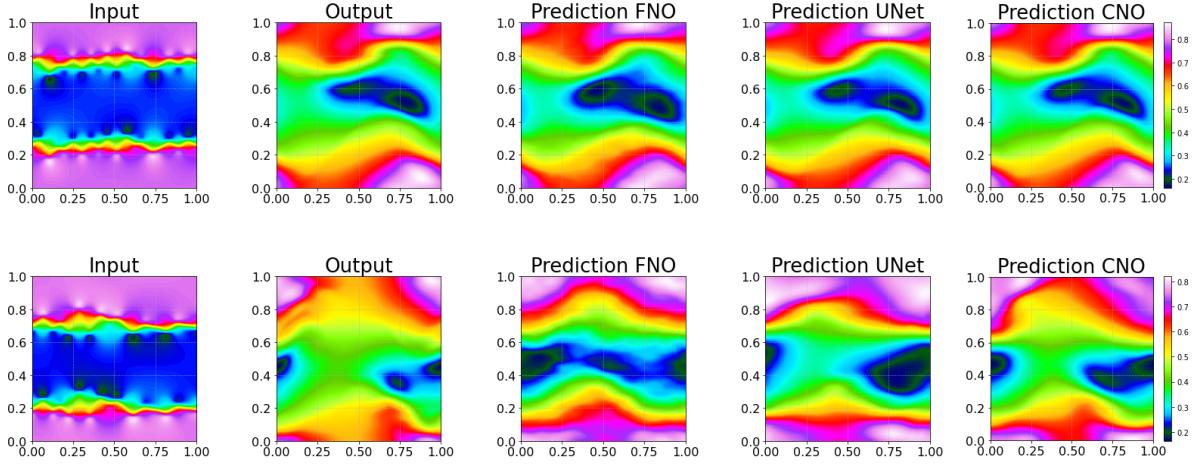


Figure 19: Navier-Stokes equations. Exact and predicted coefficients for an in-distribution (top row) and an out-of-distribution (bottom row) samples and for different models (columns). From left to right: input, output (ground truth), CNO, FNO and UNet.

### C.3.6 Darcy Flow

Steady-state Darcy flow is modeled by a PDE 4.7. The solution operator  $\mathcal{G}^\dagger : a \mapsto u$  maps the diffusion coefficient  $a$  (represented as a push forward of a Gaussian process) to the solution  $u$ . In-distribution and out-of-distribution samples differ in the length scales of the Gaussian process in 4.8. We chose the length scale  $l = 0.1$  for the in-distribution testing and  $l = 0.05$  for the out-of-distribution testing. We generate 256 training samples. In addition, we generate 128 samples for validation set, in-distribution and out-of-distribution testing. The resolution of the data is  $64 \times 64$ .

In Figure 20, we show the empirical test error distributions for UNet, FNO and CNO models during in-distribution and out-of-distribution testing. We show an in-distribution and out-of-distributions predictions made by CNO, FNO and UNet in Figure 21. The CNO model is the best-performing model in this experiment in both in-distribution and out-of-distribution testing.

### C.3.7 Flow past airfoils

The flow past the airfoil is modeled by the two-dimensional compressible Euler equations

$$u_t + \operatorname{div} F(u) = 0, \quad u = [\rho, \rho v, E]^\perp, \quad F = [\rho v, \rho v \otimes v + p\mathbf{I}, (E + p)v]^\perp, \quad (\text{C.21})$$

with density  $\rho$ , velocity  $v$ , pressure  $p$  and total Energy  $E$  related by the ideal gas equation of state:

$$E = \frac{1}{2}\rho|u|^2 + \frac{p}{\gamma - 1}, \quad (\text{C.22})$$

where  $\gamma = 1.4$ . Additional important variables associated with the flow include the speed of sound  $a = \sqrt{\frac{\gamma p}{\rho}}$  and the Mach number  $M = \frac{|u|}{a}$ .

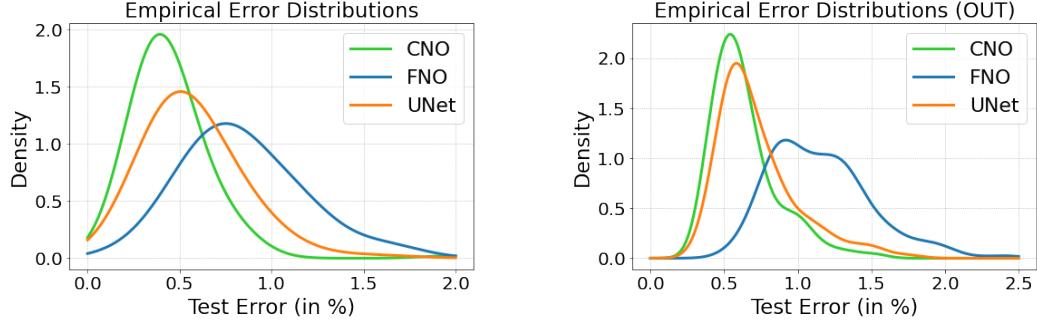


Figure 20: Darcy Flow. Empirical test error distributions for UNet, FNO and CNO. Left: In-distribution testing. Right: Out-of-distribution testing.

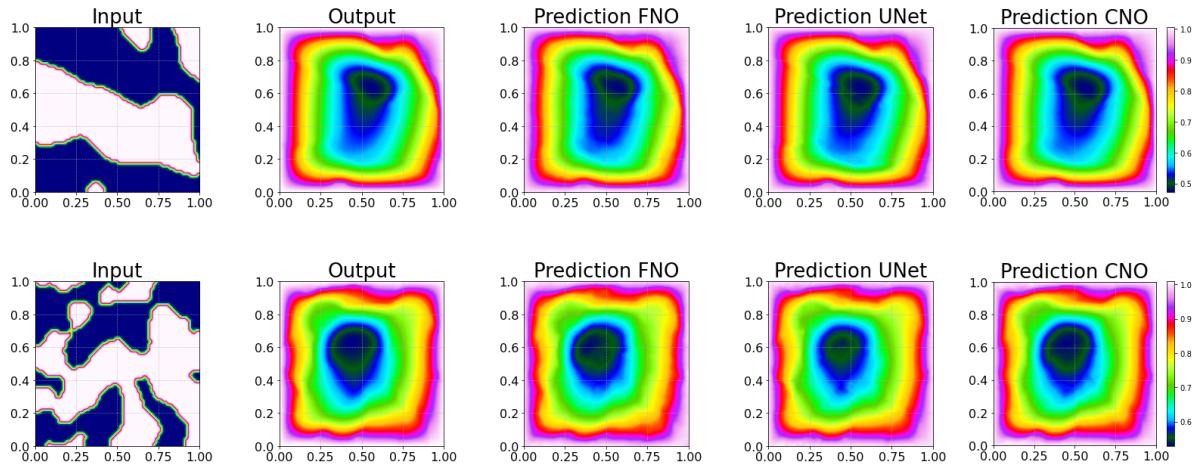


Figure 21: Darcy Flow. Exact and predicted coefficients for an in-distribution (top row) and an out-of-distribution (bottom row) samples and for different models (columns). From left to right: input, output (ground truth), FNO, UNet and CNO.

We follow standard practice in aerodynamic shape optimization and consider a reference airfoil shape with upper and lower surface of the airfoil are located at  $(x, y_{\text{ref}}^{\text{U}}(x/c))$  and  $(x, y_{\text{ref}}^{\text{L}}(x/c))$  where  $c$  is the chord length and  $y_{\text{ref}}^{\text{U}}$  and  $y_{\text{ref}}^{\text{L}}$  corresponding to the well-known RAE2822 airfoil. The reference shape is then perturbed by *Hicks-Henne Bump functions* [40] :

$$\begin{aligned} y^{\text{L}}(\xi) &= y_{\text{ref}}^{\text{L}}(\xi) + \sum_{i=1}^{10} a_i^{\text{L}} B_i(\xi), \quad y^{\text{U}}(\xi) = y_{\text{ref}}^{\text{U}}(\xi) + \sum_{i=1}^{10} a_i^{\text{U}} B_i(\xi), \\ B_i(\xi) &= \sin^3(\pi \xi^{q_i}), \quad q_i = \frac{\ln 2}{\ln 14 - \ln i}, \quad \xi = \frac{x}{c}, \\ a_i^{\text{L}} &= 2(\psi_i - 0.5)(i+1) \times 10^{-3}, \quad a_i^{\text{U}} = 2(\psi_{i+10} - 0.5)(11-i) \times 10^{-3}, \quad i = 1, \dots, 10 \end{aligned}$$

with  $\psi \in [0, 1]^d$ .

We can now formally define the airfoil shape as  $\mathcal{S} = \{(x, y) \in D : x \in [0, c], y^L \leq y \leq y^U\}$  and accordingly the shape function  $f = \chi_{[\mathcal{S}]}(x, y)$ , with  $\chi$  being the *characteristic function*. The underlying operator of interest  $\mathcal{G}^\dagger : f \mapsto \rho$  maps the shape function  $f$  into the density of the flow at steady state of the compressible Euler equations.

The equations are solved with the solver NUWTUN on  $243 \times 43$  elliptic mesh (Fig.22) given the following free-stream boundary conditions,

$$T^\infty = 1, \quad M^\infty = 0.729, \quad p^\infty = 1, \quad \alpha = 2.31^\circ.$$

The data is ultimately interpolated onto a Cartesian grid of dimensions  $128 \times 128$  on the underlying domain  $D = [-0.75, 1.75]^2$ , and unit values are assigned to the density  $\rho(x, y)$  for all  $(x, y)$  in the set  $\mathcal{S}$ .

The shapes of the training data samples correspond to 20 bump functions, with coefficients  $\psi$  sampled uniformly from  $[0, 1]^{20}$ . Out-of-distribution testing is performed with 30 bump functions. During the training and evaluation processes, the difference between the learned solution and the ground truth is exclusively calculated for the points  $(x, y)$  that do not belong to the airfoil shape  $\mathcal{S}$ .

We generate 750 samples for the training set and 128 samples for validation set, in-distribution testing set and out-of-distribution testing set. In this experiment, the data is *not* normalized. In Figure 23, we show the empirical test error distributions for UNet, FNO and CNO models during in-distribution and out-of-distribution testing. We also show a random in-distribution testing sample and an out-of-distribution testing sample, as well as predictions made by CNO, FNO and UNet in Figure 24. The latter figure clearly shows the superiority of CNO and UNet over FNO when it comes to out-of-distribution testing.

### C.3.8 On the Choice of the RPB benchmarks.

As noted in the main text, the rationale for the inclusion of benchmark experiments in the **RPB** dataset presented here is three-fold. First, we would like to span a variety of PDEs, ranging from linear elliptic (Poisson) to linear hyperbolic (wave, transport) to nonlinear parabolic (Allen-Cahn)

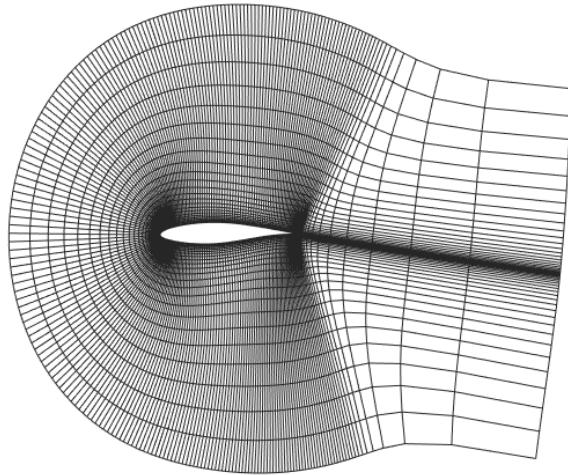


Figure 22: Elliptic mesh for the airfoil problem

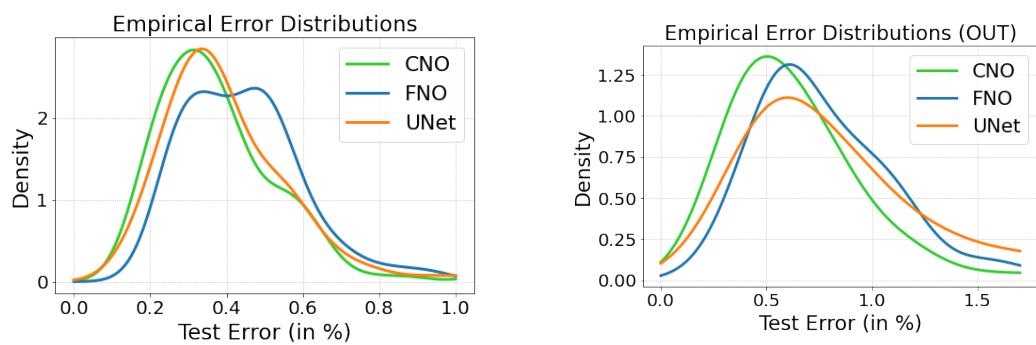


Figure 23: Airfoil experiment. Empirical test error distributions for UNet, FNO and CNO. Left: In-distribution testing. Right: Out-of-distribution testing.

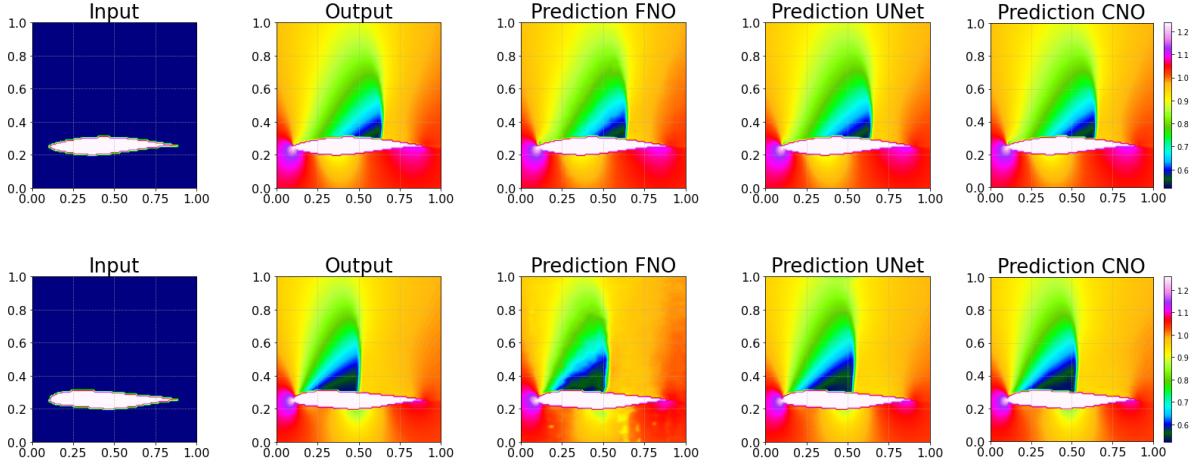


Figure 24: Airfoil experiment. Exact and predicted coefficients for an in-distribution (top row) and an out-of-distribution (bottom row) samples and for different models (columns). From left to right: input, output (ground truth), CNO, FNO and UNet.

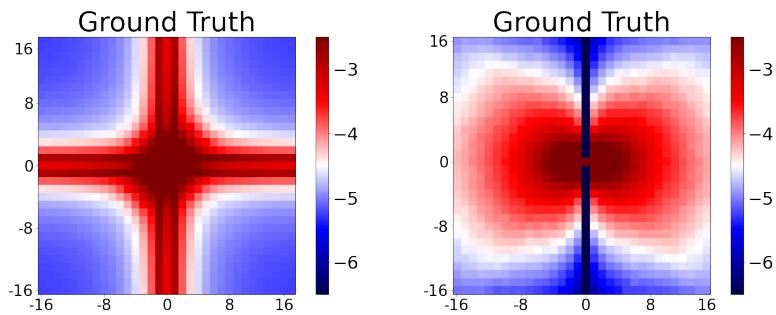


Figure 25: Comparison of the 32 central frequencies of averaged logarithmic amplitude spectra for the two Navier-Stokes experiments. Left: Old NS experiment. Right: Thin shear layer experiment.

to nonlinear hyperbolic (Compressible Euler) to non-local advection-diffusion (Incompressible Navier-Stokes). Second, we would like the underlying data to be readily available for rapid prototyping and reproducibility. This limits the use of three-dimensional data-sets as data access can be cumbersome. This requirement also leads us to prioritize problems with available analytical solutions. Finally, the selected benchmarks should be *sufficiently computationally complex* such that traditional numerical methods for approximating them are expensive and there is a potential pay-off for the design of efficient machine learning based surrogates. This criterion rules out one-dimensional (in space) problems as traditional numerical methods are very fast in this case on modern computers and there is little reason to discard them for ML surrogates. Even among two-dimensional problems, one has to be careful in selecting appropriate benchmarks to ensure that they entails sufficient computational complexity.

We illustrate this issue by comparing and contrasting two possible benchmarks. First, we consider a *Navier-Stokes* data-set, considered in [33] and widely used in the recent literature on machine learning for PDEs. In this problem, the incompressible Navier-Stokes equations (4.6) are recast in the so-called *velocity-vorticity* formulation by considering the vorticity  $\omega = \nabla \times u$  of the fluid. In two space dimensions, the following evolution equation for the vorticity can be readily derived from (4.6),

$$\omega_t + (u \cdot \nabla)\omega = \nu \Delta \omega, \quad \omega(0, \cdot) = \omega_0. \quad (\text{C.23})$$

We consider the above evolution of the vorticity with periodic boundary conditions. The underlying solution operator maps the initial vorticity  $\omega_0$  to the vorticity  $\omega(\cdot, T)$  at a final time  $T$ . Following [33], we choose the initial conditions  $\omega_0 \sim \mu$  where  $\mu = \mathcal{N}(0, 7^{\frac{3}{2}}(-\Delta + 49\text{Id})^{-2.5})$  and extend (C.23) with a forcing term  $f(x) = 0.1(\sin(2\pi(x_1 + x_2)) + \cos(2\pi(x_1 + x_2)))$ . Furthermore, the viscosity is chosen to be  $\nu = 10^{-3}$ . To generate the training and test data, we use a spectral method such as the one suggested in [27] and references therein. A rough estimate on the computational complexity of this problem can already be formed by observing Figure 25 (Left) where we present the averaged logarithmic amplitude spectra corresponding to the ground truth output (vorticity at time  $T = 30$  as considered in [33]). We clearly see from this figure that only very few frequency modes (2-3) in each direction have relatively high amplitude and the spectrum decays quite fast for higher frequencies. Thus, this problem could be potentially approximated to high accuracy on fairly coarse grids.

To provide a quantitative elaboration of the above argument, we write  $u_i^{N_f} = \mathcal{P}_{N_f}(u_i)$  where  $u_i$  is the solution corresponding to the  $i$ -th drawn initial conditions and  $\mathcal{P}_N$  is the spatial Fourier projection operator mapping a function  $f(x, t)$  to its first  $N$  Fourier modes:  $\mathcal{P}_N = \sum_{|k|_\infty \leq N} \widehat{f}_k(t) e^{ik \cdot x}$ . For each sample  $u_i$  we compute the relative  $L^1$  error against the downsampled solution  $u_i^{N_f}$ . This provides us with an estimate how many Fourier modes need to be accurately approximated in order to achieve reasonable errors. The supremum and median of the errors over 128 samples, at time  $T = 30$ , are plotted in Figure 26. One can observe from this figure that even after  $t = 30$  time units, only a maximum of 20 Fourier modes (in each direction) are needed to approximate the solution with an error of approximately 1%. Hence, a standard numerical method would only need to simulate it on a grid of  $20 \times 20$  points will suffice in

order to achieve the same error. Consequently, the time requirements for solving the problem on very coarse mesh with traditional spectral or finite difference methods are in the range of  $10^{-3}$  seconds or lower. In contrast, we tested both FNO and CNO on this dataset to obtain test errors of 1.15% and 0.96%, respectively. Moreover, the inference time for both FNO and CNO in this case are of the order of  $10^{-4}$  secs on a NVIDIA quadro t2000 GPU. Thus to achieve similar test errors, FNO and CNO are almost only one order of magnitude faster than a traditional numerical method. Given the training time and data generation overheads, it is clear that there is very little payoff on using such a relatively simple two-dimensional problem as a benchmark for ML surrogates for PDEs.

On the other hand, we perform exactly the same analysis for the *thin shear layer* problem for the incompressible Navier-Stokes equation that is described in the main text. First, from Figure 25 (Right), we see that the ground truth output (horizontal velocity at time  $T = 1$ ) has much more of a multiscale structure than in the previous experiment (compare with Figure 25 (Left)) with at least non-trivial frequencies upto 32 modes, suggesting that it is much more challenging to approximate it numerically. This is indeed verified from Figure 26 (Right) where we present the averaged (over 128 samples)  $L^1$ -error for the velocity as a function of the number of modes to observe that almost 100 Fourier modes are needed to get an  $L^1$ -error of 2%. This corresponds to a  $100 \times 100$  spatial grid and even a state-of-the-art GPU implementation of the spectral viscosity method of [27] would require  $10^{-1}$  seconds of run time. When compared to a CNO inference time of  $10^{-4}$  secs for an error of approximately 3%, we see that the ML surrogate (CNO) provides *three orders of magnitude* or more of speedup in this case, making its deployment worthwhile. Thus, we have demonstrated the rationale for the choice of this benchmark, rather than the Navier-Stokes benchmark of [33], in our proposed **RPB** dataset.

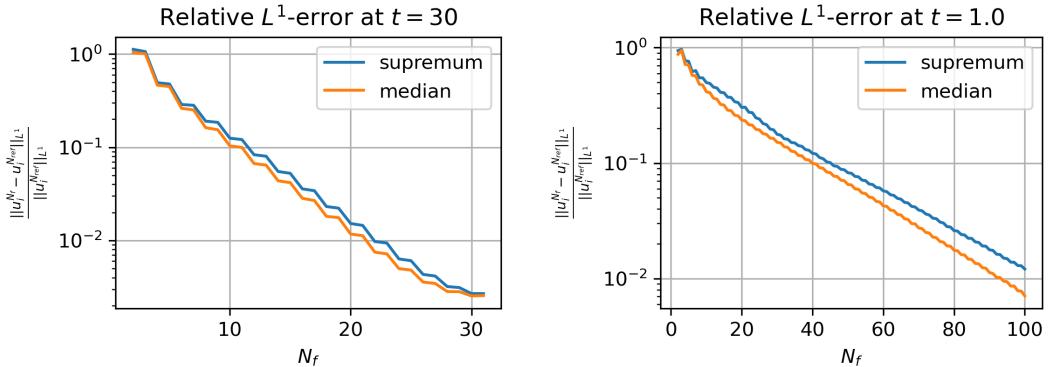


Figure 26: Relative  $L^1$ -error of the vorticity experiment when restricting the solution to  $N_f$  Fourier modes.

#### C.4 Testing at Different Resolutions.

We have emphasized repeatedly that CNO upholds the principle of continuous-discrete equivalence (CDE), which implies that there is an equivalence between the underlying operator

and its discrete representations. As a reminder, the CNO models are operators denoted as  $\mathcal{G}^* : \mathcal{B}_w(D) \rightarrow \mathcal{B}_w(D)$  and are designed to ensure that the continuous representations of functions align with their discrete samples on a uniform grid. This holds true when the sampling rate  $s$  of the grid is sufficiently high, specifically  $s \geq 2w$ . It is important to note that the implemented CNO models are specified on a predefined *computational grid* with a sampling rate of  $s \geq 2w$ . Hence following [2] Remark 3.7, the input functions must be *compatible* with this grid. If the input function is not compatible with the computational grid, one needs transform it to an appropriate representation. Once the model is applied, the output is transformed back to the original representation (see Remark 3.5 of [2] for a formal explanation) and also Formula (A.7) for a precise description of these transformations.

Hence to apply an implemented CNO model to a continuous function  $f \in \mathcal{B}_{w'}(D)$ , it is necessary to employ a discrete representation of the function on a computational grid with a sampling rate of  $s$ . Essentially, it means that one needs to sample  $f$  on that grid. If the band limit  $w'$  exceeds half the sampling rate  $s/2$ , it is crucial to first filter out frequencies above  $s/2$  to prevent aliasing effects, which involves applying a downsampling filter. Once the function's representation and the computational grid are compatible with each other, the model can be applied.

To apply an implemented CNO model to a discrete representation  $f_{s'} \in \mathbb{R}^{s' \times s'}$ , it is necessary to follow (A.7) and transform  $f_{s'}$  into a compatible representation  $f_s \in \mathbb{R}^{s \times s}$ . If  $s' \leq s$ , the signal needs to be upsampled to the sampling rate  $s$  by using an appropriate upsampling filter. However, if  $s' > s$ , it is necessary to filter out frequencies above  $s/2$  that are present in the signal. One should downsample the signal to the sampling rate  $s$  by applying an appropriate downsampling filter.

As highlighted in the main text, an important characteristic of an operator learning model is to maintain a relatively consistent test error when evaluated on various resolutions or discretizations. To assess this aspect, we evaluate the performance of UNet, FNO, and CNO models on different resolutions for Navier-Stokes equations. The original data is generated at a resolution of  $128 \times 128$ . To obtain data at any lower resolution  $s < 128$ , we downsample the original data to the desired resolution. The models that we use to make predictions are the ones that we trained on  $64 \times 64$  resolution. The configurations of all the models are reported in C.2.

We apply the afore-described strategy to practically realize Formula (A.7) and apply CNO to different resolutions. In contrast, we follow the approach outlined in [33] to evaluate the FNO or UNet models at different resolutions by applying the underlying model directly to the original, unresized input.

We show the variations of the test errors across resolutions for the Navier-Stokes benchmark in Figure 2, right. The CNO model demonstrates the highest stability when it comes to resolution changes and is (approximately) invariant to resolution, unlike the other two models which exhibit notable fluctuations at different resolutions. Specifically, the UNet model displays a strong reliance on the training resolution, whereas the FNO model exhibits a slightly less pronounced dependence. This example show that the CNO model respects continuous-discrete equivalence, while the other two models are not resolution (or representation) equivalent.

### C.5 Ablation Studies.

We conduct two ablation studies focusing on two key aspects of CNO. Firstly, we examine the impact of modified operations, assessing how they affect the overall performance. Secondly, we investigate the influence of ResNets that connect the Encoder and Decoder components within the Operator UNet architecture (refer to Figure 1). These studies aim to provide valuable insights into the effects of these key elements.

In our first ablation study, we aim to evaluate the effects of modifying operations, including upsampling operators, downsampling operators, and activation layers, on performance and training time. It is worth reiterating that the modified operations enable *continuous-discrete equivalence* (CDE). Specifically, we replace the upsampling operator in the Operator UNet architecture with a discrete, nearest neighbor upsampling method, while the downsampling operator is substituted with average pooling. Additionally, we replace the activation layer with a simple pointwise application of the activation function. As a result, the model takes on a structure resembling a regular UNet architecture, but with the inclusion of additional ResNets that establish connections between the Encoder and the Decoder components. We will refer to this model as *CNO w/o Filters*.

The second ablation study focuses on evaluating the influence of additional ResNets that connect the Encoder and the Decoder components on both the overall performance and training time. In this study, we remove these ResNets while retaining the UNet-like concatenations between corresponding levels of the Encoder and the Decoder. It is important to note that the ResNet between the deepest levels of the Encoder and the Decoder is preserved within the model. This ablation model respects the *continuous-discrete equivalence* (CDE).

**Performance.** We train two ablation models for every benchmark experiment that we studied in the main text. In order to maintain consistency, we use the same hyperparameter configurations for the ablation models as those of the best-performing CNO models (refer to Table 12 for the specific values). We report the in-distribution and out-of-distribution test errors in the Table 13.

Among the 16 tests conducted, the original CNO model outperforms the others in 12 of them. In other 4 cases, the testing errors are close to the testing errors of the best performing models. In almost all of the tests conducted, the first ablation model exhibits inferior performance compared to the original CNO model. This observation indicates that the aliasing errors resulting from regular CNN operations like average pooling, nearest neighbor upsampling and a regular application of the activation function have an impact on the test error. Furthermore, it is important to note that the first ablation study does not adhere to the continuous-discrete equivalence (CDE) property, resulting in the model’s resolution dependence, similar to the UNet model (see Figure 2 and Section C.4).

In one out-of-distribution tests, the second ablation model demonstrates slightly superior performance compared to the original CNO model. It is worth noting that in many cases,

the original CNO model exhibits significantly better performance than the second ablation model. This disparity in performance ranges from less than 10% in the Compressible Euler and Darcy Flow benchmarks to almost 300% in the Poisson Equation benchmark. While it is true that the second ablation model maintains the continuous-discrete equivalence (CDE) property, we observe that the inclusion of ResNets is vital for achieving good performance and decent generalization.

**Training time.** Since the first ablation model does not utilize any interpolation filters, it is reasonable to anticipate that it will have a faster training time than the original CNO model. Specifically, it trains approximately 1.75 times faster for the Poisson Equation, while for the Darcy Flow, it trains around 1.35 times faster. For the Navier-Stokes Equations, the training is 1.25 times faster. For the Wave Equation, Continuous Transport and Allen-Cahn Equation, it trains approximately 1.1 times faster. Finally, for the Discontinuous Transport, they need approximately equal amount of time to train.

The second ablation model, which excludes the middle ResNets from the architecture, is also expected to have faster training process than the original CNO model. Specifically, it trains approximately 1.75 times faster for the Poisson Equation. For the Navier-Stokes Equations and Darcy Flow, the model trains 1.25 times faster. The training for the Wave Equation, Discontinuous Transport and Compressible Euler is 1.1 to 1.15 times faster. Other benchmarks need similar amount of time to train.

### C.6 Error vs. number of training samples.

Once again, we revisit the best-performing CNO and FNO model architectures for the Poisson equation and Wave equation, as reported in C.2. This time, we focus on varying the number of training samples and retraining the selected CNO and FNO models accordingly. Consequently, we generate a plot that illustrates the in-distribution test error as we change the cardinality of the training set, as shown in Figure 27 (left for Poisson equation, right for Wave equation). In the case of Poisson equation, the CNO model outperforms by far the FNO model in all the data regimes. In the case of Wave equation, we notice that the FNO performs better than the CNO in low data regime, with the opposite behaviour in the large data regime. Moreover, CNO shows an approximately error decay rate of 0.5 with respect to the number of training samples.

## D Depiction of the Datasets.

In the following figures, we illustrate the different PDE forward problems considered in the main text.

Table 13: Relative median  $L^1$  test errors, for both in- and out-of-distribution testing, for the CNO models and two ablation models.

	In/Out	CNO	CNO w/o Filters	CNO w/o ResNets
<b>Poisson Equation</b>	In	<b>0.21%</b>	0.93%	0.85%
	Out	<b>0.27%</b>	1.65%	0.82%
<b>Wave Equation</b>	In	0.63%	<b>0.59%</b>	1.64%
	Out	1.17%	<b>1.12%</b>	1.64%
<b>Smooth Transport</b>	In	<b>0.24%</b>	0.31%	0.31%
	Out	<b>0.46%</b>	<b>0.46%</b>	0.76%
<b>Discontinuous Transport</b>	In	<b>1.03%</b>	1.21%	1.17%
	Out	<b>1.18%</b>	1.32%	1.60%
<b>Allen-Cahn</b>	In	<b>0.54%</b>	0.69%	0.71%
	Out	2.23%	<b>2.16%</b>	2.21%
<b>Navier-Stokes</b>	In	<b>2.76%</b>	3.20%	3.00%
	Out	7.04%	9.60%	<b>5.85%</b>
<b>Darcy</b>	In	<b>0.38%</b>	0.47%	0.41%
	Out	<b>0.50%</b>	0.65%	0.58%
<b>Compressible Euler</b>	In	<b>0.35%</b>	0.38%	0.37%
	Out	<b>0.59%</b>	0.62%	<b>0.59%</b>

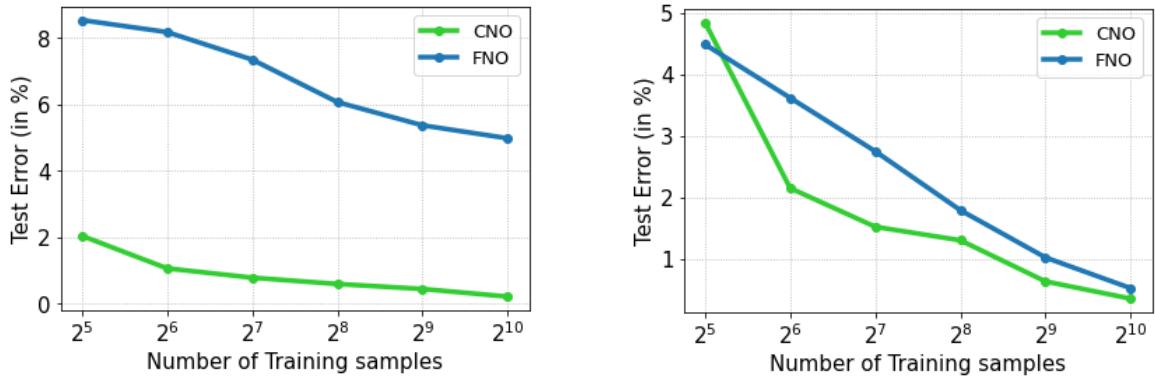


Figure 27: In-distribution testing errors for different cardinalities of the training set for FNO and CNO. Left: Poisson equation. Right: Wave equation.

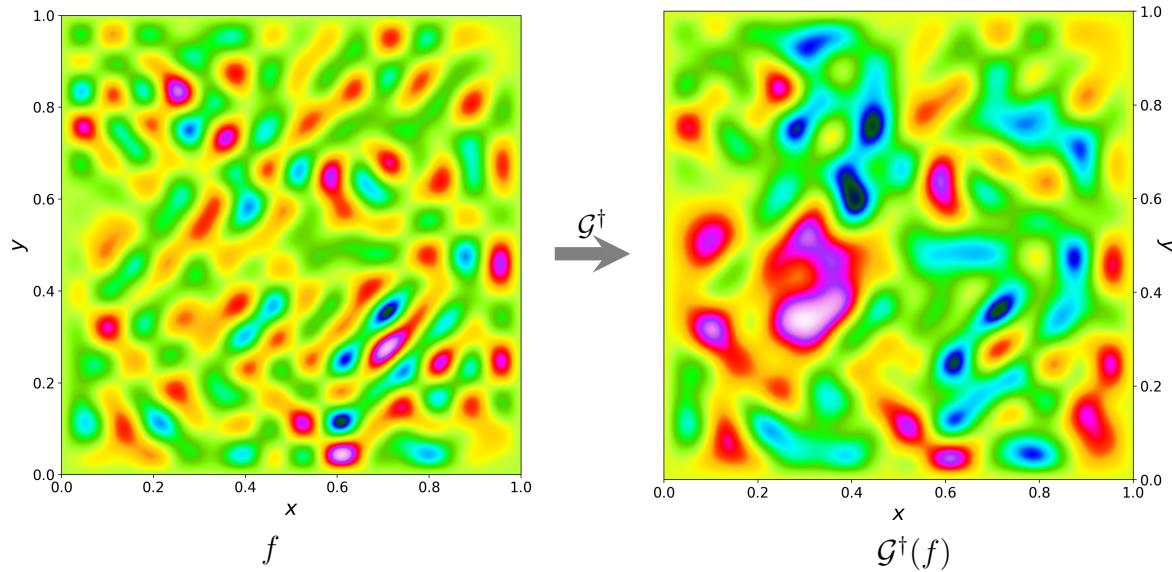


Figure 28: Illustration of input (left) and output (right) samples for the Poisson Equation.

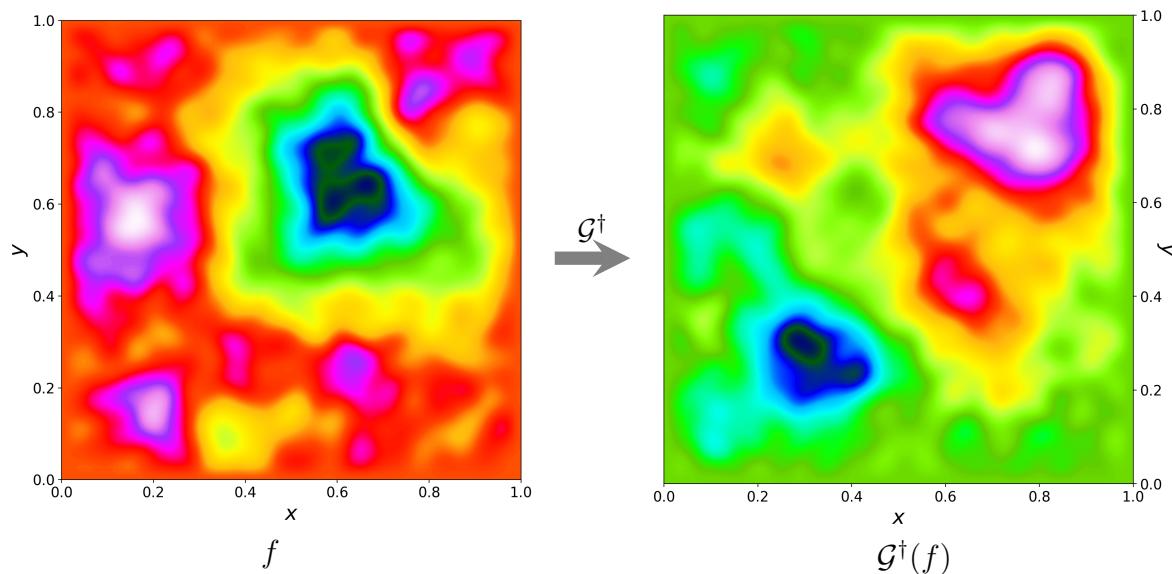


Figure 29: Illustration of input (left) and output (right) samples for the Wave Equation.

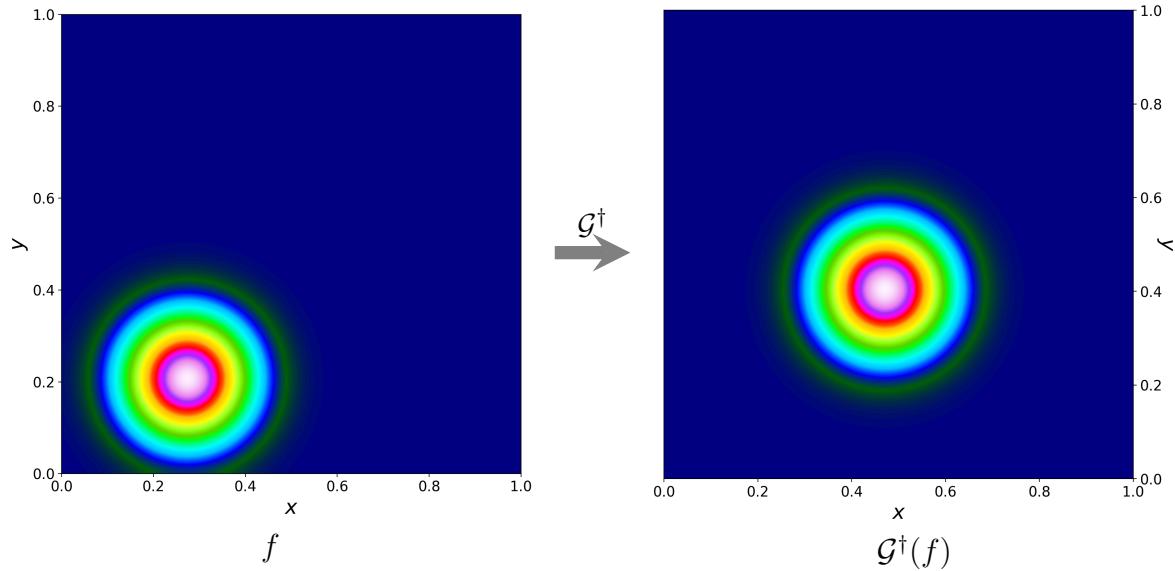


Figure 30: Illustration of input (left) and output (right) samples for the Continuous Transport.

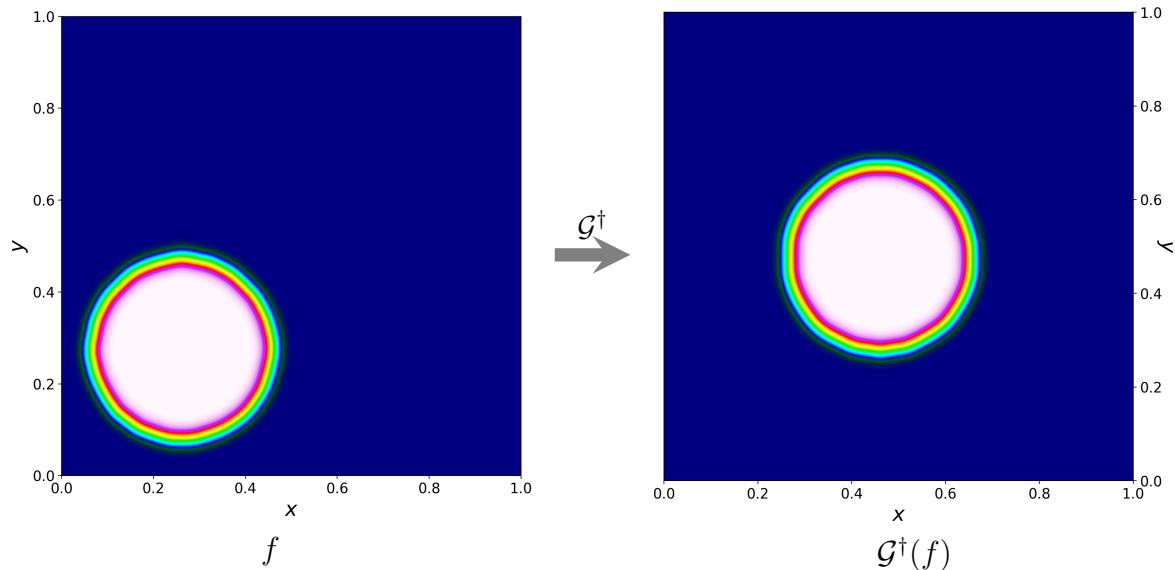


Figure 31: Illustration of input (left) and output (right) samples for the discontinuous transport problem.

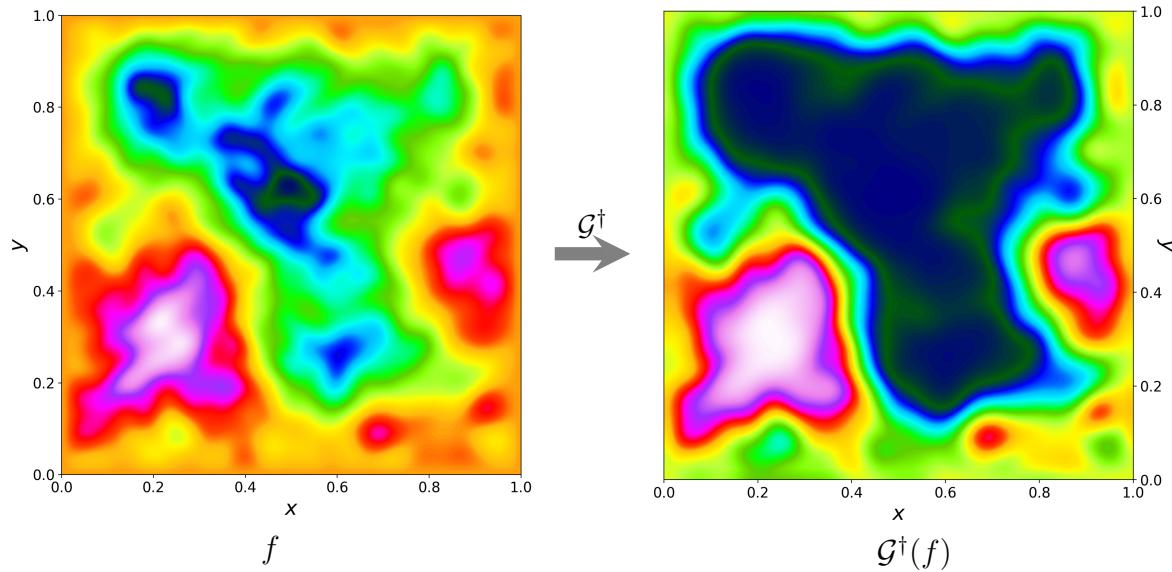


Figure 32: Illustration of input (left) and output (right) samples for the Allen-Cahn equation.

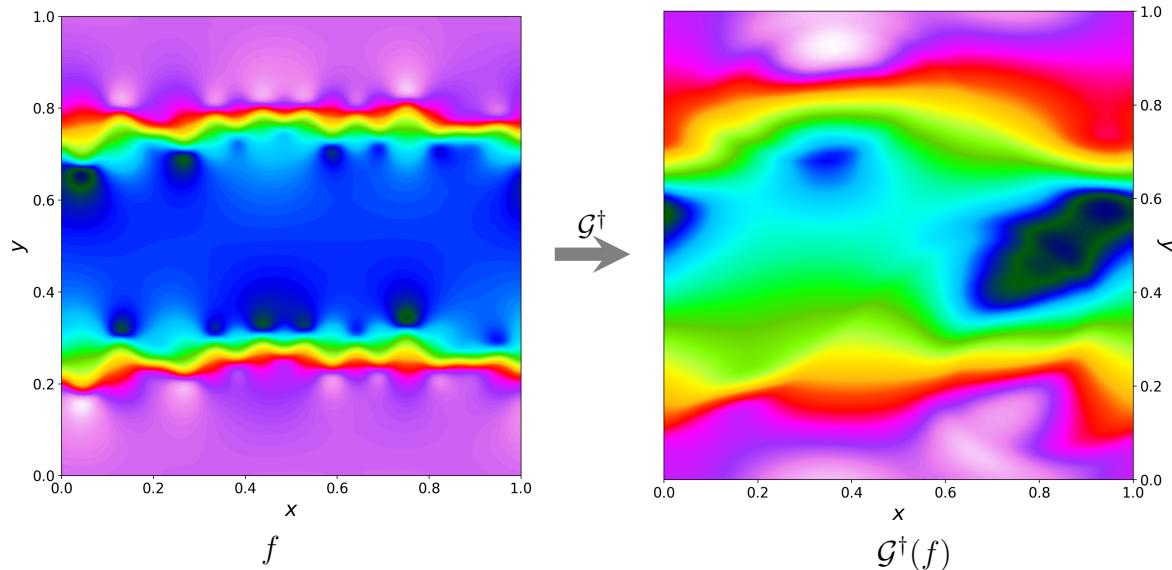


Figure 33: Illustration of input (left) and output (right) samples for the Navier-Stokes equations.

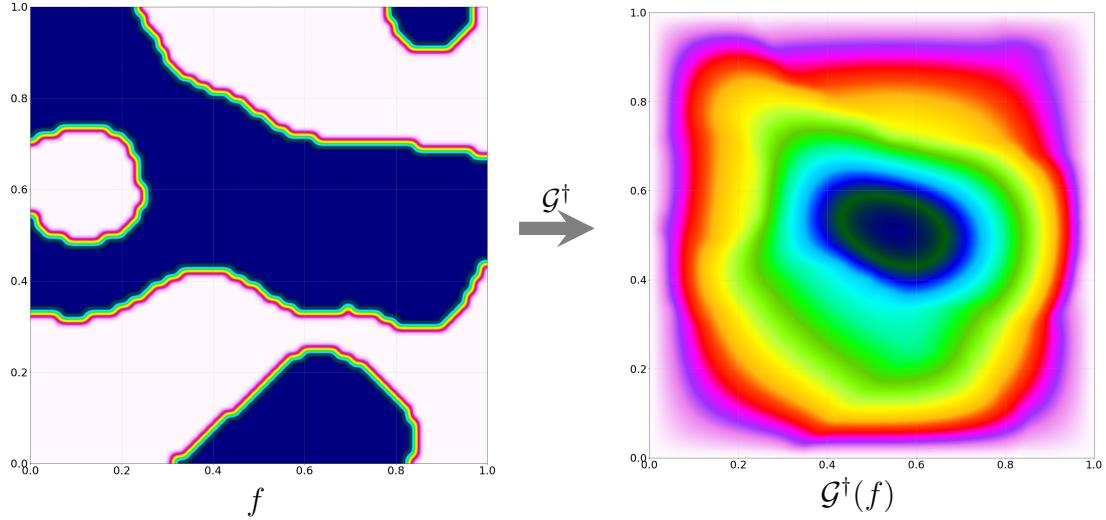


Figure 34: Illustration of input (left) and output (right) samples for the Darcy flow.

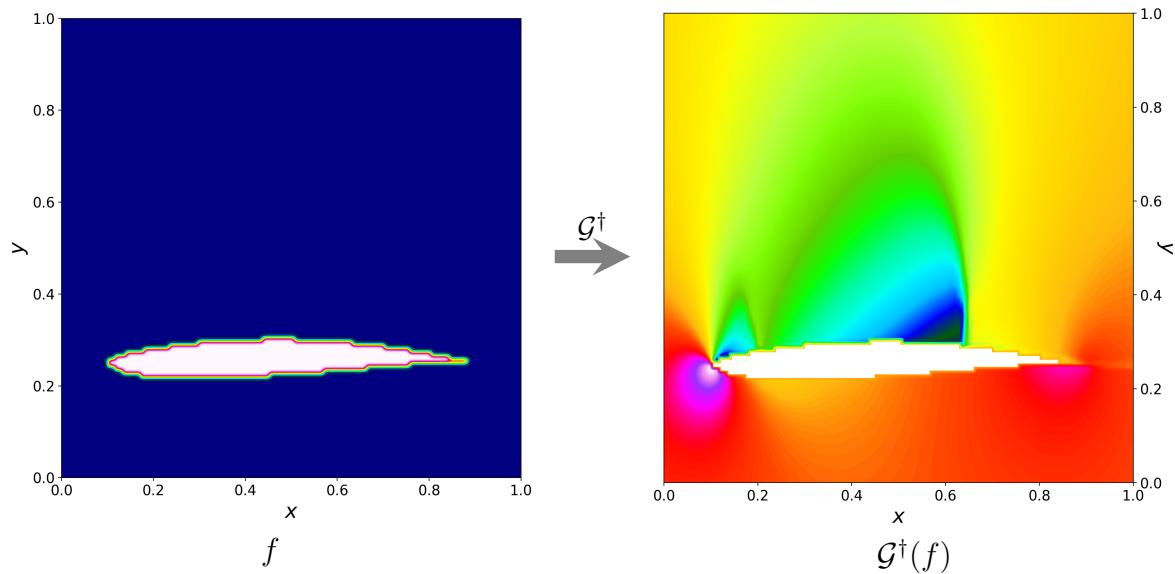


Figure 35: Illustration of input (left) and output (right) samples for the compressible Euler equations.

## References

- [1] I. Ayed, E. de Bézenac, A. Pajot, J. Brajard, and P. Gallinari. Learning dynamical systems from partial observations. *CoRR*, abs/1902.11136, 2019.
- [2] F. Bartolucci, E. de Bézenac, B. Raonic, R. Molinaro, S. Mishra, and R. Alaifari. Representation equivalent neural operators: a framework for alias-free operator learning. *arXiv:2305.19913*, 2023.
- [3] J. Bell, P. Collela, and H. M. Glaz. A second-order projection method for the incompressible Navier-Stokes equations. *J. Comput. Phys.*, 85:257–283, 1989.
- [4] M. Bertero, P. Bocacci, and C. De Mol. *Introduction to inverse problems in imaging*. CRC press, 2021.
- [5] K. Bhattacharya, B. Hosseini, N. B. Kovachki, and A. M. Stuart. Model Reduction And Neural Networks For Parametric PDEs. *The SMAI journal of computational mathematics*, 7:121–157, 2021.
- [6] N. Boullé, Y. Nakatsukasa, and A. Townsend. Rational neural networks. *Advances in Neural Information Processing Systems*, 33:14243–14253, 2020.
- [7] S. Cai, Z. Wang, L. Lu, T. A. Zaki, and G. E. Karniadakis. DeepM&Mnet: Inferring the electroconvection multiphysics fields based on operator approximation by neural networks. *Journal of Computational Physics*, 436:110296, 2021.
- [8] S. Cao. Choose a transformer: Fourier or galerkin. In *35th conference on neural information processing systems*, 2021.
- [9] T. Chen and H. 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.
- [10] T. De Ryck, S. Lanthaler, and S. Mishra. On the approximation of functions by tanh neural networks. *Neural Networks*, 2021.
- [11] T. De Ryck and S. Mishra. Generic bounds on the approximation error for physics-informed (and) operator learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.
- [12] Q. Delfosse, P. Schramowski, M. Mundt, A. Molina, and K. Kersting. Adaptive rational activations to boost deep reinforcement learning. *arXiv preprint arXiv:2102.09407*, 2021.
- [13] L. C. Evans. *Partial differential equations*, volume 19. American Mathematical Soc., 2010.
- [14] V. Fanaskov and I. Oseledets. Spectral neural operators. *arXiv preprint arXiv:2205.10573v1*, 2022.

- [15] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio. *Deep learning*, volume 1. MIT Press, 2016.
- [16] J. K. Gupta and J. Brandstetter. Towards multi-spatiotemporal-scale generalized pde modeling, 2022.
- [17] E. Haber and L. Ruthotto. Stable architectures for deep neural networks. *Inverse problems*, 34, 2018.
- [18] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *arXiv: 1512.03385*, 2015.
- [19] J. S. Hesthaven, S. Gottlieb, and D. Gottlieb. *Spectral methods for time-dependent problems*, volume 21. Cambridge University Press, 2007.
- [20] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. Image-to-image translation with conditional adversarial networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2017.
- [21] G. E. Karniadakis, I. G. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, and L. Yang. Physics informed machine learning. *Nature Reviews Physics*, pages 1–19, may 2021.
- [22] T. Karras, M. Aittala, S. Laine, E. Häkkinen, J. Hellsten, J. Lehtinen, and T. Aila. Alias-free generative adversarial networks. *Advances in Neural Information Processing Systems*, 34:852–863, 2021.
- [23] G. Kissas, J. H. Seidman, L. F. Guilhoto, V. M. Preciado, G. J. Pappas, and P. Perdikaris. Learning operators with coupled attention. *Journal of Machine Learning Research*, 23(215):1–63, 2022.
- [24] N. Kovachki, S. Lanthaler, and S. Mishra. On universal approximation and error bounds for fourier neural operators. *Journal of Machine Learning Research*, 22:Art–No, 2021.
- [25] N. Kovachki, Z. Li, B. Liu, K. Azizzadenesheli, K. Bhattacharya, A. Stuart, and A. Anandkumar. Neural operator: Learning maps between function spaces. *arXiv preprint arXiv:2108.08481v3*, 2021.
- [26] S. Lanthaler, S. Mishra, and G. E. Karniadakis. Error estimates for DeepONets: A deep learning framework in infinite dimensions. *Transactions of Mathematics and Its Applications*, 6(1):tnac001, 2022.
- [27] S. Lanthaler, S. Mishra, and C. Parés-Pulido. Statistical solutions of the incompressible euler equations. *Mathematical Models and Methods in Applied Sciences*, 31(02):223–292, Feb 2021.
- [28] S. Lanthaler, R. Molinaro, P. Hadorn, and S. Mishra. Nonlinear reconstruction for operator learning of pdes with discontinuities. In *International Conference on Learning Representations*, 2023.

- [29] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.
- [30] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.
- [31] M. Leshno, V. Y. Lin, A. Pinkus, and S. Schocken. Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. *Neural networks*, 6(6):861–867, 1993.
- [32] Z. Li, D. Z. Huang, B. Liu, and A. Anandkumar. Fourier neural operator with learned deformations for pdes on general geometries, 2022.
- [33] Z. Li, N. B. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. Stuart, and A. Anandkumar. Fourier neural operator for parametric partial differential equations. In *International Conference on Learning Representations*, 2021.
- [34] Z. Li, N. B. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. M. Stuart, and A. Anandkumar. Neural operator: Graph kernel network for partial differential equations. *CoRR*, abs/2003.03485, 2020.
- [35] Z. Li, N. B. Kovachki, K. Azizzadenesheli, B. Liu, A. M. Stuart, K. Bhattacharya, and A. Anandkumar. Multipole graph neural operator for parametric partial differential equations. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, pages 6755–6766. Curran Associates, Inc., 2020.
- [36] Z. Li, H. Zheng, N. Kovachki, D. Jin, H. Chen, B. Liu, K. Azizzadenesheli, and A. Anandkumar. Physics-informed neural operator for learning partial differential equations. *arXiv preprint arXiv:2111.03794*, 2021.
- [37] Z. Liu, H. Mao, C. Wu, C. Feichtenhofer, T. Darrell, and S. Xie. A convnet for the 2020s. In *Proceedings - 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022*, Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pages 11966–11976. IEEE Computer Society, 2022.
- [38] Z. Long, Y. Lu, X. Ma, and B. Dong. Pde-net: Learning pdes from data. In J. G. Dy and A. Krause, editors, *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 3214–3222. PMLR, 2018.
- [39] L. Lu, P. Jin, G. Pang, Z. Zhang, and G. E. Karniadakis. Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators. *Nature Machine Intelligence*, 3(3):218–229, 2021.
- [40] K. O. Lye, S. Mishra, D. Ray, and P. Chandrashekhar. Iterative surrogate model optimization (ISMO): An active learning algorithm for PDE constrained optimization with deep neural networks. *Computer Methods in Applied Mechanics and Engineering*, 374:113575, 2021.

- [41] Z. Mao, L. Lu, O. Marxen, T. Zaki, and G. E. Karniadakis. DeepMandMnet for hypersonics: Predicting the coupled flow and finite-rate chemistry behind a normal shock using neural-network approximation of operators. Preprint, available from arXiv:2011.03349v1, 2020.
- [42] D. A. Masters, N. J. Taylor, T. Rendall, C. B. Allen, and D. J. Poole. Geometric comparison of aerofoil shape parameterization methods. *AIAA Journal*, pages 1575–1589, 2017.
- [43] A. Molina, P. Schramowski, and K. Kersting. Pad\`e activation units: End-to-end learning of flexible activation functions in deep networks. *arXiv preprint arXiv:1907.06732*, 2019.
- [44] R. Molinaro, y. Yang, E. Engquist, and S. Mishra. Neural inverse operators for solving pde inverse problems. *arXiv:2301.11167*, 2023.
- [45] J. Pathak, S. Subramanian, P. Harrington, S. Raja, A. Chattopadhyay, M. Mardani, T. Kurth, D. Hall, Z. Li, K. Azizzadenesheli, p. Hassanzadeh, K. Kashinath, and A. Anandkumar. Fourcastnet: A global data-driven high-resolution weather model using adaptive fourier neural operators. *arXiv preprint arXiv:2202.11214*, 2022.
- [46] P. Petersen and F. Voigtlaender. Equivalence of approximation by convolutional neural networks and fully-connected networks. *Proceedings of the American Mathematical Society*, 148(4):1567–1581, 2020.
- [47] M. Prasthofer, T. De Ryck, and S. Mishra. Variable input deep operator networks. *arXiv preprint arXiv:2205.11404*, 2022.
- [48] A. Quarteroni and A. Valli. *Numerical approximation of Partial differential equations*, volume 23. Springer, 1994.
- [49] 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, 2019.
- [50] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention-MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*, pages 234–241. Springer, 2015.
- [51] T. Schanze. Sinc interpolation of discrete periodic signals. *IEEE Transactions on Signal Processing*, 43(6):1502–1503, 1995.
- [52] J. H. Seidman, G. Kissas, P. Perdikaris, and G. J. Pappas. NOMAD: Nonlinear manifold decoders for operator learning. *arXiv preprint arXiv:2206.03551*, 2022.
- [53] E. Tadmor. Convergence of spectral methods for nonlinear conservation laws. *SIAM Journal on Numerical Analysis*, 26(1):30–44, 1989.
- [54] E. Tadmor. Burgers' Equation with Vanishing Hyper-Viscosity. *Communications in Mathematical Sciences*, 2(2):317 – 324, 2004.

- [55] M. Tancik, P. Srinivasan, B. Mildenhall, S. Fridovich-Keil, N. Raghavan, U. Singhal, R. Ramamoorthi, J. Barron, and R. Ng. Fourier features let networks learn high frequency functions in low dimensional domains. *Advances in Neural Information Processing Systems*, 33:7537–7547, 2020.
- [56] M. Telgarsky. Neural networks and rational functions. In *International Conference on Machine Learning*, pages 3387–3393. PMLR, 2017.
- [57] A. Tran, A. Mathews, L. Xie, and C. S. Ong. Factorized fourier neural operators. In *The Eleventh International Conference on Learning Representations*, 2023.
- [58] M. Unser. Sampling-50 years after shannon. *Proceedings of the IEEE*, 88(4):569–587, 2000.
- [59] M. Vetterli, J. Kovacevic, and V. Goyal. *Foundations of Signal Processing*. Cambridge University Press, 2014.
- [60] S. Wang, S. Suo, W. Ma, A. Pokrovsky, and R. Urtasun. Deep parametric continuous convolutional neural networks. *CoRR*, abs/2101.06742, 2021.
- [61] S. Wang, H. Wang, and P. Perdikaris. Learning the solution operator of parametric partial differential equations with physics-informed DeepOnets. *arXiv preprint arXiv:2103.10974*, 2021.
- [62] S. E. Wei. Aliasing-free nonlinear signal processing using implicitly defined functions. *IEEE Access*, 10:76281–76295, 2022.
- [63] R. Wightman, H. Touvron, and H. Jégou. Resnet strikes back: An improved training procedure in timm. *CoRR*, abs/2110.00476, 2021.
- [64] J. Yang, Q. Du, and W. Zhang. Uniform l p-bound of the allen-cahn equation and its numerical discretization. *International Journal of Numerical Analysis & Modeling.*, 15, 2018.
- [65] Y. Zhu and N. Zabaras. Bayesian deep convolutional encoder–decoder networks for surrogate modeling and uncertainty quantification. *Journal of Computational Physics*, 336:415–447, 2018.