Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Seminar for Applied Mathematics SAM

Variable-Input Deep Operator Networks

M. Prasthofer and T. De Ryck and S. Mishra

Research Report No. 2022-19
May 2022

Seminar für Angewandte Mathematik
Eidgenössische Technische Hochschule
CH-8092 Zürich
Switzerland

Variable-Input Deep Operator Networks

Michael Prasthofer* **Tim De Ryck*** **Siddhartha Mishra**
Seminar for Applied Mathematics
ETH Zürich, Switzerland

Abstract

Existing architectures for operator learning require that the number and locations of sensors (where the input functions are evaluated) remain the same across all training and test samples, significantly restricting the range of their applicability. We address this issue by proposing a novel operator learning framework, termed Variable-Input Deep Operator Network (VIDON), which allows for random sensors whose number and locations can vary across samples. VIDON is invariant to permutations of sensor locations and is proved to be universal in approximating a class of continuous operators. We also prove that VIDON can efficiently approximate operators arising in PDEs. Numerical experiments with a diverse set of PDEs are presented to illustrate the robust performance of VIDON in learning operators.

1 Introduction

Operators are mappings between infinite-dimensional spaces. They arise in a large variety of contexts in science and engineering, particularly when the underlying models are ordinary (ODEs) or partial (PDEs) differential equations. A prototypical example for operators is provided by the so-called solution or evolution operator of a time-dependent PDE, which maps an input (infinite-dimensional) function space of initial conditions to an output function space of solutions of the PDE at certain point of time. Given the ubiquity of ODEs and PDEs in applications, *learning operators* from data is of great significance in science and engineering, [9, 13] and references therein.

As inputs and outputs of operators are infinite-dimensional (e.g. functions, infinite sequences), conventional neural networks cannot be directly deployed to learn them. Instead, a new field of *operator learning* is rapidly emerging, wherein one designs novel learning architectures to approximate such operators. A popular operator learning paradigm is that of *neural operators* [13], which generalize the structure of neural networks, wherein each hidden layer consists of a *non-local* affine operator, composed with a local (scalar) non-linear activation function. Reflecting the infinite-dimensional structure of the underlying learning task, the non-local affine layer amounts to integrating with respect to a kernel and choosing different kernels leads to graph kernel operators, [17], low-rank kernel operators [13] and multipole expansions [18]. Evaluating a convolution-based kernel efficiently in Fourier space (via FFT), yields the *Fourier neural operator* (FNO) [16], which has been rigorously proved to be *universal*, efficient in learning operators arising in PDEs [12] as well as being very successfully employed in a variety of applications in science and engineering [16, 19, 24] and references therein. However, FNOs are restricted to operators where the discretization of the underlying domains is (or can be efficiently mapped to) a Cartesian grid, considerably limiting the range of their applicability [22].

An alternative framework is that of *operator networks* [3] and their deep version, *DeepONets* [21]. This architecture is based on two different sets of neural networks, so-called *trunk nets* which span the infinite-dimensional output space and *branch nets* which map the (encoded) input into the coefficients of the trunk nets. DeepONets are also universal, efficiently approximate operators arising in PDEs

*Equal contribution.

Preprint. Under review.

[14] and are widely used in scientific computing [21, 23, 1, 20] and references therein. In contrast to FNOs, DeepONets can handle learning operators on very general domains and boundary conditions. However, DeepONets have their own set of limitations. In particular, the (infinite-dimensional) input function to the branch net of a DeepONet has to be projected to finite dimensions by evaluating it on a finite set of *sensors*, located in the underlying domain. Although these sensor locations can be *randomly chosen* inside the domain [21, 14], the number and location of these sensor points has to be invariant across all training (and test) samples. Similarly, FNOs require input sensors to be located on a Cartesian grid for each sample.

On the other hand, the training (and test) data for operator learning is generated either from physical measurements (observations) or computer simulations (or a combination of them). In both scenarios, it is too restrictive to expect that data is available at the same set (or number) of points across all training samples. For instance, different data sources (measurement devices) can be placed at different locations at different time periods or for different domains (experimental conditions), within the same data set. Moreover, measurements at some sensor locations could be missing due to device faults. Additionally, there will always be an intrinsic uncertainty in the exact location of measurement devices. Similarly, numerical simulations at different spatio-temporal resolutions will be combined in the same training and test data set. Hence, the lack of flexibility in sensor locations for *encoding* the inputs to DeepONets and FNOs constitutes a major limitation for current operator learning frameworks (see Figure 1).

This limitation of fixed number and location of sensors points to a fundamental issue with existing operator learning frameworks as one can view their inputs as vectors of fixed length containing function values at fixed locations, rather than functions which could be evaluated at arbitrary points of the domain. This raises questions on the very essence of operator learning with current architectures. A related issue pertains to the notion of *permutation invariance* i.e., permuting input sensor locations should not alter the output of an operator learning framework as the same underlying input function is being sampled. Thus, it is imperative to require that operator learning frameworks be permutation-invariant.

The above considerations set the stage for the current paper where we propose a novel architecture for operator learning that allows for variable (flexible) sensor locations. To this end and motivated by the permutation-invariant deep learning frameworks such as *deep sets* [29, 27] as well as *transformers* [26], we propose an operator learning framework termed *Variable-Input Deep Operator Networks* (VIDON) in Section 2, that allows for random locations for input sensors in each sample as well as for the number and locations of sensors to vary across samples (see Figure 1). We prove in Section 3 that VIDON is *universal* in approximating continuous operators that map into Sobolev spaces. Moreover, we also prove that VIDON can efficiently approximate operators stemming from a variety of PDEs, by showing that the size of the underlying neural networks only grows polynomially (at worst) in the inverse of the accuracy. Finally in Section 4, we illustrate VIDON in a series of numerical experiments for different PDEs, showing that it can accurately approximate the underlying operators, while presented with very different input sensor configurations. The notation and technical details for proofs and implementations are presented in the supplementary material (**SM**).

2 Variable-Input Deep Operator Networks

Setting. For compact sets $D, U \subset \mathbb{R}^d$, we consider (general forms of) operators $\mathcal{G} : \mathcal{X} \to \mathcal{Y}$, where $\mathcal{X} \subset L^2(D; \mathbb{R}^{d_v})$ and $\mathcal{Y} \subset L^2(U; \mathbb{R}^{d_u})$. In the context of time-dependent PDEs of the general abstract form, $\mathcal{L}_a(u) = 0$ with $u(0, \cdot) = u_0$, where a is a parameter function, we will consider operators that map a parameter function a to the solution u i.e., $\mathcal{G} : \mathcal{X} \to \mathcal{Y} : a \mapsto u$, as well as operators that map the initial condition u_0 to the solution u i.e., $\mathcal{G} : \mathcal{X} \to \mathcal{Y} : u_0 \mapsto u$.

DeepONets. Following [21, 14], one approximates the operators $\mathcal{G}$ by starting with an *encoder* $\mathcal{E} : \mathcal{X} \to \mathbb{R}^m$ that maps every input function u to m fixed *sensors* and its sensor values $((x_j, u(x_j)))_{1 \le j \le m}$. Two sets of neural networks are then defined, a *branch net* $\boldsymbol{\beta}^* : \mathbb{R}^m \to \mathbb{R}^p$ and a *trunk net* $\boldsymbol{\tau} : U \to \mathbb{R}^{p+1}$. The branch and trunk nets are then combined to approximate the underlying non-linear operator as the *DeepONet* $\mathcal{N}^*$,

$$\mathcal{N}^* : \mathcal{E}(\mathcal{X}) \subset \mathbb{R}^m \to \mathcal{Y} : \mathcal{E}(u) \mapsto \tau_0(y) + \sum_{k=1}^{p} \beta_k^*(\mathcal{E}(u))\tau_k(y). \tag{2.1}$$

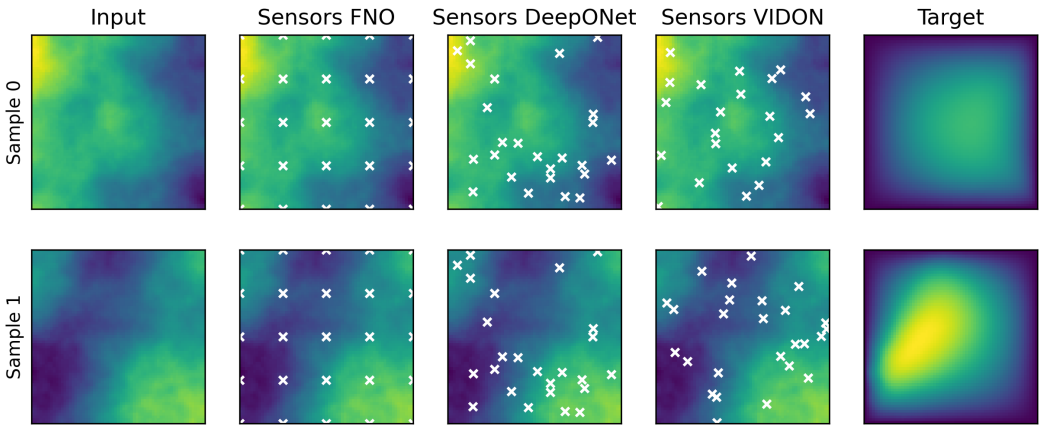


Figure 1: Learning the operator mapping the permeability coefficient a (Input) to the solution u (Target) for Darcy flow (3.3), for two different realizations (samples) of the input. The input has to be evaluated at *sensor* points. FNO requires a Cartesian grid of sensors for each sample whereas as DeepONets allow for a random cloud of sensors. However, the number and location of sensors has to be the same for all training and test samples. In contrast, VIDON (2.6) allows for random sensor points, whose location and number can vary for each sample.

Functions on sets. In DeepONets (2.1), the input to the branch net is a tuple of fixed length, whose entries are function values of the input function u at m fixed sensors. As described in the introduction, this is necessarily limiting. To allow for variable sensor locations as well as invariance of the output to permutations of sensor locations, the input to the branch net should consist of sets of variable size, rather than tuples of fixed length. To this end, we follow [27] to define $\mathfrak{X}^{\leq M}$ to be the set of subsets of a set $\mathfrak{X}$ containing at most $M \in \mathbb{N}$ elements and we denote by $\mathfrak{X}^{\mathcal{F}}$ the set of finite subsets of $\mathfrak{X}$. For $M \in \mathbb{N}$, $u \in \mathcal{X}$ and (possibly random) sensor points $x_1(u), \dots, x_M(u)$ we can then define an encoder $\mathcal{E}$ by,

$$\begin{aligned} &\mathcal{E} : \mathcal{X} \times \{1, \dots, M\} \to (\mathbb{R}^{d+d_v})^{\leq M} : (u, m) \to \mathcal{E}_m(u), \quad \text{where,} \\ &\mathcal{E}_m : \mathcal{X} \to (\mathbb{R}^{d+d_v})^m : u \mapsto \{(x_j, u(x_j))\}_{j=1}^m, \quad 1 \leq m \leq M. \end{aligned} \tag{2.2}$$

Given this encoding, the branch net in (2.1) needs to be a permutation-invariant function that allows sets with variable size as input, i.e., $\boldsymbol{\beta} : (\mathbb{R}^{d+d_v})^{\leq M} \to \mathbb{R}^p$. Based on the result of [29] on permutation-invariant functions, it has been proven in [27, Theorem 4.1] that every continuous function $f : \mathbb{R}^{\leq M} \to \mathbb{R}$ is necessarily *continuously sum-decomposable via* $\mathbb{R}^M$, meaning that it must be of the form $f(X) = \rho\left(\sum_{x \in X} \varphi(x)\right)$, $X \in \mathbb{R}^{\leq M}$, where $\rho : \mathbb{R}^M \to \mathbb{R}$ and $\varphi : \mathbb{R} \to \mathbb{R}^M$ are continuous functions. This characterization constitutes the starting point of our new architecture, which consists of the following ingredients,

Input encoding. The input is a function $u \in \mathcal{X}$ that is sampled at $m = m(u)$ sensor points by the encoder $\mathcal{E}_m$ (2.2) i.e., $\mathcal{E}_m(u) = \{(x_j, u(x_j))\}_{j=1}^m$ where $\{x_j\}_{j=1}^m \subset D$ are the sensor coordinates and $\{u(x_j)\}_{j=1}^m \subset \mathbb{R}^{d_v}$ the corresponding sensor values. Note that the sampling procedure, both in terms of number of sensors and their locations, is allowed to be different for every input. The samples are then further encoded as follows,

$$\Psi : \mathbb{R}^{d+d_v} \to \mathbb{R}^{d_{enc}} : (x_j, u(x_j)) \mapsto \Psi_c\left(x_j\right) + \Psi_v\left(u(x_j)\right) =: \psi_j \tag{2.3}$$

where $\Psi_c : \mathbb{R}^d \to \mathbb{R}^{d_{enc}}$ is the *coordinate encoder* and $\Psi_v : \mathbb{R}^{d_v} \to \mathbb{R}^{d_{enc}}$ the *value encoder*, both modeled as trainable multilayer perceptrons (MLPs).

Head(s). The well-known *attention* mechanism of the transformer [26] architecture transforms sequential inputs into weighted values, where the (convex) weights are computed in terms of *correlations* between inputs at different locations. Loosely motivated by this mechanism but seeking to keep computational complexity linear (instead of quadratic as in the case of attention of [26]) in inputs, we

process the encoded inputs $\{\psi_j\}_{j=1}^m$ in the following manner. First for any $1 \leq \ell \leq H$, the *values* of the *head* ℓ are calculated using a single MLP $\widetilde{\nu}^{(\ell)} : \mathbb{R}^{d_{enc}} \to \mathbb{R}^p$. Next, the corresponding weights are computed as,

$$\omega^{(\ell)} : \mathbb{R}^{d_{enc}} \to \mathbb{R} : \psi_j \mapsto \frac{\exp\left(\widetilde{\omega}^{(\ell)}(\psi_j)/\sqrt{d_{enc}}\right)}{\sum_{k=1}^m \exp\left(\widetilde{\omega}^{(\ell)}(\psi_k)/\sqrt{d_{enc}}\right)}, \qquad \text{where} \quad \widetilde{\omega}^{(\ell)} : \mathbb{R}^{d_{enc}} \to \mathbb{R} \tag{2.4}$$

is instantiated as an MLP. The output of a single head with index ℓ is then given by,

$$\nu^{(\ell)} : \mathbb{R}^{d_{enc}} \to \mathbb{R}^p : \Psi(\mathcal{E}_m(u)) = (\psi_j)_{j=1}^m \mapsto \sum_{j=1}^m \omega^{(\ell)}(\psi_j)\widetilde{\nu}^{(\ell)}(\psi_j). \tag{2.5}$$

Note that ν^ℓ is permutation-invariant and well-defined for any $m \in \mathbb{N}$. Next, in analogy to multihead attention [26], we concatenate the multiple heads $\nu^{(\ell)}$ and denote the result by $\nu = [\nu^{(1)}, \ldots, \nu^{(H)}]$.

Variable-Input Deep Operator Network. Finally, we combine the outputs of the multiple heads using another MLP $\Phi : \mathbb{R}^{H \cdot p} \to \mathbb{R}^p$. The Variable-Input Deep Operator Network (VIDON) is then defined by replacing the branch net $\boldsymbol{\beta}^*$ in (2.1) by $\boldsymbol{\beta} := \Phi \circ \nu \circ \Psi$ i.e., for any $m \in \mathbb{N}$ we define,

$$\mathcal{N} : \mathcal{E}(\mathcal{X}) \to \mathcal{Y} : \mathcal{E}_m(u) \mapsto \tau_0(y) + \sum_{k=1}^p \beta_k(\mathcal{E}_m(u))\tau_k(y). \tag{2.6}$$

We measure the size of VIDON, $\text{size}(\mathcal{N})$, by the total number of unique parameters in both the branch and trunk nets, which is independent of the number of sensors (**SM** Remark B.1). The structure of VIDON is illustrated and summarized in Figure 2. Moreover, by construction, the new branch net $\boldsymbol{\beta}$ in VIDON (2.6) is a permutation-invariant function that takes finite sets of variable sizes as input.

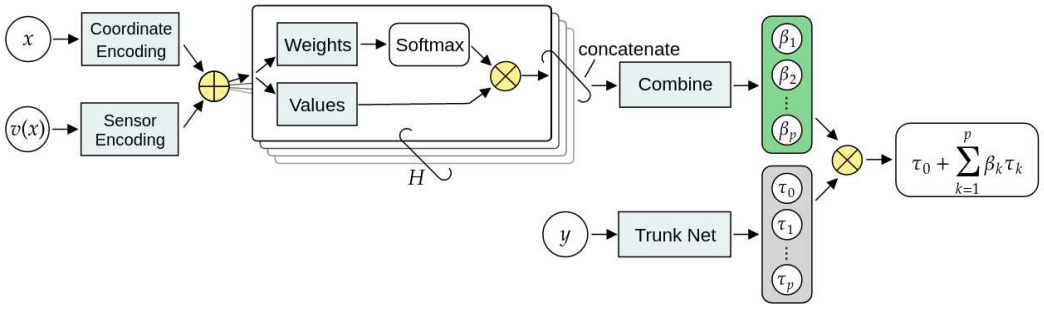


Figure 2: Structure of VIDON (2.6). The coordinates and values at each sensor are encoded through MLPs and are processed through another set of MLPs to compute weights and values. The resulting convex combination of the outputs from different sensors constitutes the output of a single head. Multiple heads are concatenated and combined through another MLP to yield the branch net, which is then combined with the trunk net to obtain the VIDON output. Light blue shading represents a MLP.

Related work. Our proposed architecture, VIDON (2.6), is related to the DeepONet operator learning framework of [21] (see [3] for the original (shallow) version of operator networks). As in the case of DeepONets, the underlying operator is approximated by VIDON with branch and trunk nets. While the structure of trunk nets is identical in both architectures, the branch net of VIDON is different as it allows variable number of sensors at variable locations. In particular, the branch net of VIDON is permutation-invariant. Moreover, one can recover DeepONets (2.1) as a special case of VIDON (2.6) by setting $H = m$, $\Psi_c \equiv 0$, $\Psi_v \equiv \text{Id}$, $\widetilde{\omega}_k^{(\ell)} = \delta_{k\ell}$ and $\widetilde{\nu}^{(\ell)} \equiv \text{Id}$. Just like DeepONets, VIDONs can also be viewed as a neural operator by slightly modifying the construction proposed in [13, Section 3.2]. Moreover, given the fact that FNOs can viewed as DeepONets with a fixed Fourier basis for trunk nets and sensors located on Cartesian grids [12, Theorem 36], fixing the Fourier basis for trunk nets of VIDON recovers FNO. However, FNO is only limited to sensors located at Cartesian grid points. Compared to both DeepONets and FNOs, VIDON's main advantage lies in its ability

to handle input functions with both a variable number of sensor points as well as different sensor locations for each sample.

Given it's designed to be invariant with respect to permutations of sensor locations, VIDON is related to deep learning architectures that approximate functions on sets. These include *deep sets* [29, 27] as well as Neural Processes [6, 5], and Attentive Neural Processes [10]. However, the branch net in VIDON has a more general structure, based on weighted sums of inputs (2.6).

Finally, there are connections between VIDON and the extensive literature on transformers. To start with, the idea of positional encoding (2.3) is borrowed from the field of sequence modelling. Instead of the popular choice of fixing the encoding (e.g. a sinusoidal encoding in the transformers of [26]), we learn the optimal encoding using a MLP in VIDON. Another relevant transformer-based architecture is the *Set Transformer* [15], which uses permutation-invariant attention blocks to learn functions on sets. However, the main difference between transformer based architectures and VIDON lies in the fact that the *attention* head in transformers accounts for correlations between the different coordinates, whereas the *head* (2.5) in VIDON does not require any interactions between different coordinates and only assigns a weight for each coordinate based on its intrinsic value. As a result, VIDON scales as $\mathcal{O}(m)$ instead of $\mathcal{O}(m^2)$ (as standard transformers do), with m being the number of sensors. It is also worth mentioning recent works that use transformer-type architectures such as the Fourier and Galerkin transformers of [2] as well as the coupled attention-based LOCA framework of [11]. Although these architectures are shown to perform well on numerical experiments, they do not possess the rigorous theoretical guarantees nor the flexibility of inputs of VIDON.

3 Rigorous analysis of Variable-Input Deep Operator Networks

Universal approximation theorem. Conventional neural networks are universal in the sense that they can approximate any continuous function. In similar vein, one can show that operator learning frameworks such as DeepONets (in [14]) and FNOs (in [12]) are also *universal* in being able to, in principle, approximate any continuous operator. As a first step in the rigorous analysis of VIDON, we prove (**SM** B.1) the following universal approximation theorem for VIDON,

Theorem 3.1. *Let $\mathcal{G} : \mathcal{X} \to H^s(U)$ be an α-Hölder continuous operator, let μ be a measure on $L^2(D)$ whose covariance operator has a bounded eigenbasis with eigenvectors $\{\lambda_j\}_{j\in\mathbb{N}}$ and let $\mathcal{E} : \mathcal{X} \to (\mathbb{R}^{d+d_v})^{\leq M}$ for $M \in \mathbb{N}$ be a random encoder i.e.,* (2.2) *with the underlying sensor points being randomly drawn from the uniform distribution on D. Then for every $p \in \mathbb{N}$, there exists a VIDON $\mathcal{N} : \mathbb{R}^{\leq M} \to H^s(U)$* (2.6)*, with p branch and trunk nets such that for every $m \in \mathbb{N}$ with $m \leq M$ it holds with probability 1 that,*

$$\left\|\mathcal{G}(u_0) - \mathcal{N}(\mathcal{E}(u_0))\right\|_{L^2(\mu)} \lesssim \left(\sum\nolimits_{j>m/C\log(m)} \lambda_j\right)^{\alpha/2} + p^{-s/d}, \tag{3.1}$$

where the constant $C > 0$ only depends on $\mathcal{G}$ and μ.

Thus, any Hölder-continuous operator that maps into a Sobolev space can be approximated by a VIDON to desired accuracy. These assumptions are satisfied by a wide variety of operators arising in ODEs and PDEs as seen in the following. To illustrate the quantitative error bound in (3.1), we readily apply (3.1) to the often encountered example of Gaussian measures (see for instance [14, Section 3.5.1] for definitions) to obtain,

Corollary 3.2. *If the measure μ in the statement of Theorem 3.1 is Gaussian, then* (3.1) *reduces to*

$$\left\|\mathcal{G}(u_0) - \mathcal{N}(\mathcal{E}(u_0))\right\|_{L^2(\mu)} \lesssim \exp\left(-Cm^2/\log(m)^2\right) + p^{-s/d}. \tag{3.2}$$

Hence for Gaussian measures, we see that the error decays exponentially in the number of sensors.

Computational complexity of VIDON. The universal approximation theorem (Theorem 3.1) shows that one can find a VIDON such that the approximation error for the underlying operator can be made as small as needed, as long as one chooses m and p to be large enough. Although this result provides information about the required number of sensors m and number of branch and trunk nets p to obtain a certain accuracy, it does not reveal how large the underlying networks should be.

Given the underlying infinite-dimensional nature of the operator, one observes that the network size of both DeepONets [14, Remark 3.2] and FNOs [12, Remark 22] can grow (super)exponentially with respect to the desired accuracy, while approximating general continuous operators. By leveraging the relation between VIDON and DeepONets, one can translate these results to show that the needed size of VIDON (see **SM** Remark B.1 for definitions) to approximate certain operators $\mathcal{G}$ to an accuracy $\varepsilon > 0$ may grow as $\varepsilon^{-\xi(\varepsilon)}$, where $\xi : (0, \infty) \to \mathbb{N}$ is a monotonically decreasing function (see **SM** Remark B.2 for details). Such exponential growth clearly inhibits efficient approximation of operators. However, in [14] and [12], it was shown that in the special case of operators arising in a wide variety of PDEs, DeepONets and FNOs, respectively, can mitigate this exponential growth in size as one can prove that the corresponding network size will only grow *polynomially* with respect to the error. We will prove similar *efficient approximation* results for VIDON. To do so, we follow [14, 12] and consider several prototypical examples of operators arising in PDEs, defined on the d-dimensional torus $D = \mathbb{T}^d = [0, 2\pi)^d$. This should however not be seen as a restriction, as for every Lipschitz domain D with $\overline{D} \subset \mathbb{T}^d$ there exists a (continuous and linear) periodic extension operator $\mathfrak{E} : W^{m,p}(D) \to W^{m,p}(\mathbb{T}^d)$, $m \in \mathbb{N}, 1 \leq p \leq \infty$, such that $\mathfrak{E}(u)|_D = u$ and $\mathfrak{E}(u)$ and its derivatives are $\mathbb{T}^d$-periodic [12, Lemma 41].

Darcy-type elliptic PDE. We start with a standard elliptic PDE that models the steady-state pressure for a fluid flowing in a porous medium according to the Darcy's law, or diffusion of heat in a material with variable thermal conductivity. Given $\mathcal{A}^\ell_\lambda(\mathbb{T}^d) = \{a \in H^\ell(\mathbb{T}^d) \mid \|a\|_{H^\ell(\mathbb{T}^d)} \leq \lambda^{-1},\ \|a - 1\|_{L^\infty(\mathbb{T}^d)} \leq 1 - \lambda\}$ for $\lambda > 0$ and $\ell \in \mathbb{N}$, we consider the operator $\mathcal{G} : \mathcal{A}^\ell_\lambda(\mathbb{T}^d) \to \dot{H}^1(\mathbb{T}^d)$, $a \mapsto u$, where u solves the Darcy equation

$$-\nabla \cdot (a \nabla u) = f, \quad \fint_{\mathbb{T}^d} u(x)\, dx = 0, \tag{3.3}$$

on the periodic torus $\mathbb{T}^d$, with right-hand side $f \in \dot{H}^{k-1}$. For this nonlinear operator that maps the coefficients of a PDE to its solution, we have the following result on its approximation by VIDON,

Theorem 3.3. *Let $\ell > d$, $\lambda \in (0, 1)$ and let $\mathcal{E}$ be a random encoder. There exists a VIDON $\mathcal{N} : (\mathbb{R}^{d+1})^{\mathcal{F}} \to H^1(\mathbb{T}^d)$ (2.6), with p trunk nets such that for every $a \in \mathcal{A}^\ell_\lambda(\mathbb{T}^d)$ and $m \geq p^{1+\ell/d}$,*

$$\mathbb{E}\left[\left\|\mathcal{N}(\mathcal{E}_m(a)) - \mathcal{G}(a)\right\|_{L^2(\mathbb{T}^d)}\right] \leq C(p^{1-\ell/d} + p^2 m^{-1/2}). \tag{3.4}$$

It holds that $\operatorname{depth}(\boldsymbol{\beta}) = \mathcal{O}(\log(p))$, $\operatorname{width}(\boldsymbol{\beta}) = \mathcal{O}(p^{(d+1)/d})$, $\operatorname{size}(\boldsymbol{\beta}) = \mathcal{O}(p^3)$, $\operatorname{depth}(\boldsymbol{\tau}) = 2$, $\operatorname{width}(\boldsymbol{\tau}) = \mathcal{O}(p^{(d+1)/d})$ *and* $\operatorname{size}(\boldsymbol{\tau}) = \mathcal{O}(p^{(d+2)/d})$ *In particular, to obtain an accuracy of $\varepsilon > 0$ in* (3.4)*, it suffices that* $\operatorname{size}(\mathcal{N}) = \mathcal{O}(\varepsilon^{-3d/(\ell-d)})$ *and* $m = \mathcal{O}(\varepsilon^{-2(\ell+d)/(\ell-d)})$.

We sketch the proof here, while deferring the details to **SM** B.2. First, we approximate the coefficient a by a truncated version of its Fourier series and then further approximate the underlying Fourier coefficients based on the random encoder $\mathcal{E}_m(a)$ using a Monte-Carlo approach (**SM** Lemma A.6). Emulating a pseudospectral numerical scheme for the Darcy equation by neural networks as in [12, Theorem 3.5] and approximating the Fourier basis by neural networks (**SM** Lemma A.5) then gives rise to a suitable approximation of the operator $\mathcal{G}(a)$. We observe from the above theorem that, given an error tolerance ε, the size of VIDON and the number of sensors only grow polynomially in ε.

Nonlinear parabolic PDE: Allen-Cahn equation. Next, we consider a nonlinear time-dependent *parabolic PDE*, the so-called Allen-Cahn equation, which models reaction-diffusion phenomena with phase separations and transitions,

$$\partial_t u = \Delta_x u + \frac{u(u^2 - 1)}{\varepsilon^2}, \qquad u(t = 0) = u_0 \tag{3.5}$$

where u is the state and $\varepsilon > 0$ is a parameter that controls the contribution of the nonlinear term. We are interested in approximating the solution or time-evolution operator $\mathcal{G} : C^\ell(\mathbb{T}^d) \to C^\ell(\mathbb{T}^d) : u_0 \mapsto u(T)$, mapping initial conditions u_0 to solutions at final time T. We have the following theorem (proved in **SM** B.3) on the *efficient* approximation of this operator with VIDON,

Theorem 3.4. *Let $p, \ell \in \mathbb{N}$ with $\ell \geq 4$ and let $\mathcal{E}$ be a random encoder. There exists a VIDON $\mathcal{N} : (\mathbb{R}^{d+1})^{\mathcal{F}} \to C(\mathbb{T}^d)$ (2.6), with p trunk nets such that for every $u_0 \in C^\ell(\mathbb{T}^d)$ and $m \in \mathbb{N}$,*

$$\mathbb{E}\left[\|\mathcal{G}(u_0) - \mathcal{N}(\mathcal{E}_m(u_0))\|_{L^2(\mathbb{T}^d)}\right] \leq C(p^{-\ell/d} + p^{3(\ell+d)/(\ell-d)} m^{-1/2}). \tag{3.6}$$

It holds that $\mathrm{depth}(\boldsymbol{\beta}) = \mathcal{O}(p^{1+\ell/d})$, $\mathrm{width}(\boldsymbol{\beta}) = \mathcal{O}(p^{(d+\ell)/2+1/d})$, $\mathrm{size}(\boldsymbol{\beta}) = \mathcal{O}(p^{(d+\ell)/2+2/d})$, $\mathrm{depth}(\boldsymbol{\tau}) = 2$, $\mathrm{width}(\boldsymbol{\tau}) = \mathcal{O}(p^{(d+1)/d})$ *and* $\mathrm{size}(\boldsymbol{\tau}) = \mathcal{O}(p^{(d+2)/d})$. *In particular, to obtain an accuracy of* $\varepsilon > 0$ *in* (3.6)*, it suffices that* $\mathrm{size}(\mathcal{N}) = \mathcal{O}(\varepsilon^{-d(d+\ell)/2\ell-2\ell})$ *and* $m = \mathcal{O}(\varepsilon^{-2-6d(\ell+d)/\ell(\ell-d)})$.

Navier-Stokes equations. The motion of a viscous, incompressible Newtonian fluid is modeled by the well-known incompressible Navier-Stokes equations,

$$\partial_t u + u \cdot \nabla u + \nabla p = \nu \Delta u, \quad \mathrm{div}(u) = 0,\ u(t=0) = u_0. \tag{3.7}$$

Here, $u \in \mathbb{R}^d$ is the velocity and $p \in \mathbb{R}$ is the pressure of the fluid, ν the viscosity and the initial velocity is denoted by u_0. For simplicity, we assume periodic boundary conditions in the domain $\mathbb{T}^d$ and consider solutions of (3.7) that are in the set $\mathcal{V} \subset C([0,T]; H^r) \cap C^1([0,T]; H^{r-2})$ (see **SM** Definition B.3 for notation). Writing $\mathcal{V}_t := \{u(t) \mid u \in \mathcal{V}\}$, we want to approximate the solution operator $\mathcal{G}$ of (3.7), mapping initial data $u_0 = u(t=0)$, to the solution $u(T)$ at $t = T$. We have the following theorem (proved in **SM** B.4) on *efficient* approximation of this operator with VIDON,

Theorem 3.5. *Let* $p \in \mathbb{N}$, $r \geq d/2 + 2$ *and let* $\mathcal{E}$ *be a random encoder. There exists a Variable-Input Deep Operator Network* $\mathcal{N} : (\mathbb{R}^{d+1})^{\mathcal{F}} \to H^1(\mathbb{T}^d)$ *with* p *trunk nets such that for every* $u_0 \in \mathcal{V}_0$ *and* $m \in \mathbb{N}$*,*

$$\mathbb{E}\left[\|\mathcal{G}(u_0) - \mathcal{N}(\mathcal{E}_m(u_0))\|_{L^2(\mathbb{T}^d)}\right] \leq C(p^{-r/d} + pm^{-1/2}). \tag{3.8}$$

It holds that $\mathrm{depth}(\boldsymbol{\beta}) = \mathcal{O}(\log(p))$, $\mathrm{width}(\boldsymbol{\beta}) = \mathcal{O}(p^{(d+1)/d})$, $\mathrm{size}(\boldsymbol{\beta}) = \mathcal{O}(p^3)$, $\mathrm{depth}(\boldsymbol{\tau}) = 2$, $\mathrm{width}(\boldsymbol{\tau}) = \mathcal{O}(p^{(d+1)/d})$ *and* $\mathrm{size}(\boldsymbol{\tau}) = \mathcal{O}(p^{(d+2)/d})$. *In particular, to obtain an accuracy of* $\varepsilon > 0$ *in* (3.4)*, it suffices that* $\mathrm{size}(\mathcal{N}) = \mathcal{O}(\varepsilon^{-3d/r})$ *and* $m = \mathcal{O}(\varepsilon^{-2-2d/r})$.

4 Experiments

Setting. We will compare the performance of VIDON to baselines, DeepONets [21] and FNOs [16]. As the main point of VIDON is the flexibility with respect to the number of sensors and their locations, we will consider the following configurations (with increasing levels of difficulty) in each experiment (see **SM** Figure 3 for illustrations). *Regular Grid*: The sensors are located on a regular Cartesian grid, with equally spaced grid sizes in each direction, on the domain D. This is an idealized situation (particularly for training data based on physical measurements) and rarely holds in practice. *Irregular Grid*: Here, the sensor locations are on a fixed non-Cartesian grid for all training and test samples. *Missing Data.* To model missing sensor data, we randomly delete a certain proportion (up to 20%) of sensor locations from the original Cartesian grid. *Perturbed Grid*: We model both missing data as well as uncertainty in sensor locations by randomly perturbing the original sensor locations (on a regular Cartesian Grid) and deleting up to 20% of them. *Random Sensor Locations.* Here, although the number of sensors is fixed across samples, their location is chosen randomly for each sample. *Variable Random Locations.* We randomly vary both the number (with a variance of up to 10%) as well as the location of sensors, across samples, in this configuration.

Given this lineup of configurations, FNO can only be evaluated for the idealized configuration (*Regular Grid*), DeepONets can be evaluated for *Regular Grid* and *Fixed Irregular Grid* whereas VIDON is designed to handle every single configuration described above. For each of these configurations, our training set contains 1000 samples whereas the test set has 5000 samples. The details of the training and hyperparameters for each configuration and architecture is provided in **SM** C.1

Darcy flow. We consider the Darcy-type elliptic PDE (3.3) on the domain $D = [0,1]^2$ with periodic boundary conditions. The corresponding operator $\mathcal{G}$ maps the coefficient a to the solution u of (3.3). We choose the permeability coefficient a for training (and test) samples from the underlying measure $\mathbf{N}(0, \mathbf{C})$, i.e. a Gaussian measure with a covariance operator $\mathbf{C}$, given by $\mathbf{C} = (-\Delta + 9I)^{-2}$. See [16] for details on generating such coefficients and Figure 1 (leftmost column) for two examples. The corresponding solutions u are then computed with a centered finite-difference scheme on a 251^2 Cartesian grid (see rightmost column in Figure 1 for examples) and constitute the ground truth for both training and testing. The resulting mean test error for each configuration listed above is presented in Table 1 (see **SM** Table 6 for the corresponding standard deviations). We observe from this table that for the idealized configuration of a regular grid, FNO outperforms both DeepONet and

Table 1: Mean relative Test errors in $L^2(D)$ for the Darcy flow problem for different configurations of sensors. The symbol "-" implies that the model could not be used for this configuration.

Configuration	# (Sensors)	**FNO**	**DeepONet**	**VIDON**
Regular Grid	51×51	0.76%	1.48%	1.29%
Irregular Grid	$51^2 = 2601$	-	1.52%	1.48%
Missing Data	$[2081, 2601]$	-	-	1.77%
Perturbed Grid	[2341,2861]	-	-	1.68%
Random Locations	2601	-	-	2.58%
Variable Random Locations	[2341,2861]	-	-	2.55%

VIDON. The fact that FNO can yield superior performance to DeepONet on problems where both frameworks apply is well-studied [13] and even theoretically investigated [12]. As VIDON is related to DeepONets, we expect a similar performance of both architectures which is borne out on both the regular grid as well as the fixed irregular grid, where DeepONets and VIDON can be evaluated but FNO cannot be used. The main advantage of VIDON lies in the fact that it can be applied to the other sensor configurations where neither DeepONet nor FNO work. In all these configurations, we observe that VIDON approximates the underlying operator for Darcy flow at errors which are marginally higher than the error for the idealized configuration of sensors located on a regular grid. As expected, the error with VIDON increases with increasing difficulty of the configurations while always remaining under 2.6% for this particular problem. In particular, the highest amplitude of error is realized for the *random* and *variable random* sensor locations. This is completely expected as in these configurations, the sensor locations are randomly drawn and with the maximum number of sensors being limited to 2861, there could be gaps of significant area, where no sensor information is available. Nevertheless, VIDON was able to approximate the underlying operator with slightly larger error (less than a factor of 2) over the idealized configuration of a regular grid.

Table 2: Mean relative Test errors in $L^2(D \times (0,T))$ for the Allen-Cahn PDE for different configurations of sensors. The symbol "-" implies that the model could not be used for this configuration.

Configuration	# (Sensors)	**DeepONet**	**VIDON**
Regular Grid	26×26	0.34%	0.26%
Irregular Grid	$26^2 = 676$	0.34%	0.27%
Missing Data	$[541, 676]$	-	0.63%
Perturbed Grid	[608,744]	-	0.83%
Random Locations	676	-	1.21%
Variable Random Locations	[608,744]	-	1.20%

Allen-Cahn equation. Next, we consider the Allen-Cahn equation (3.5) on the spatial domain $D = [0, 2]^2$, with initial data $u_0 = u(x, y, 0)$ that corresponds to the *rotated travelling wave*,

$$u(x, y, t) = \frac{1}{2} - \frac{1}{2} \tanh\left(\frac{1}{2\sqrt{2}\varepsilon} \begin{bmatrix} c_x \\ c_y \end{bmatrix} \cdot \begin{bmatrix} x - o_x \\ y - o_y \end{bmatrix} - \frac{3t}{\sqrt{2}\varepsilon} \right), \quad c_x^2 + c_y^2 = 1. \tag{4.1}$$

The PDE is supplemented with zero-Neumann boundary conditions, in the rotated frame. We approximate the operator $\mathcal{G}$ that maps the initial condition u_0 to the entire time-evolution $u(x, y, t)$, for all $t \in [0, T]$. The training (and test) initial conditions are generated by sampling the parameters $\varepsilon \in [0, 13, 0.18], o_x, o_y \in [0, 2], c_x \in [0, 1]$ uniformly. See **SM** Figure 4 for examples of the input and output of the operator $\mathcal{G}$. Training and test data are extracted from the exact solution. VIDON is compared with baseline DeepONet here. The standard form of FNO [16] cannot be used to approximate the operator $\mathcal{G}$ mapping the initial condition to the entire time-history. Although a recurrent version of FNO can be used instead [16], we chose not to do so as it can only be evaluated at fixed time steps and is thus not directly comparable to the non-reccurent versions of DeepONet and VIDON. The mean errors for all the configurations of sensor locations are presented in Table 2 (standard deviations are presented in **SM** Table 7) and are completely consistent with those observed for the Darcy flow problem (Table 1). The only difference being that the amplitude of test errors

is significantly lower than that of the Darcy flow. This is explained by the fact that the underlying operator is easier to learn in this case.

Navier-Stokes equations. In the final numerical experiment, we consider the incompressible Navier-Stokes equations (3.7), with viscosity $\nu = 10^{-3}$ on the spatial domain $D = [0, 1]^2$ with periodic boundary conditions. We choose the initial condition from the underlying measure $\mathbf{N}(0, \mathbf{C})$ with covariance operator $\mathbf{C} = 7^{3/2}(-\Delta + 49I)^{-2.5}$, using the code from [16]. The training (and test) data are generated with a spectral method in space and a third-order Runge-Kutta method in time [7] that approximates the fluid velocity and pressure. The resulting vorticity (curl of the velocity) can be readily computed. We seek to approximate the operator $\mathcal{G}$ that maps the initial vorticity to the vorticity at a final time $T = 5$, see **SM** Figure 5 for examples of the input-output for this operator. The results for VIDON (and baselines) for different sensor configurations are presented in Table 3 (see standard deviations in **SM** Table 8) and are qualitatively consistent with those obtained for the Darcy flow and Allen-Cahn equation. The main difference being that the amplitude of the error for both VIDON as well as the baselines is larger, corresponding to the fact that the underlying operator is harder to learn. Nevertheless, VIDON can learn with comparable accuracy to baselines at the idealized configuration, the operator for both missing data as well as perturbed sensor locations. The error is higher for random and variable random locations, but less than a factor of 2 over the DeepONet baseline for the regular grid.

Table 3: Mean relative Test errors in $L^2(D)$ for the Navier-Stokes PDE for different configurations of sensors. The symbol "-" implies that the model could not be used for this configuration.

Configuration	# (Sensors)	**FNO**	**DeepONet**	**VIDON**
Regular Grid	33×33	3.49%	4.20%	5.22%
Irregular Grid	$33^2 = 1089$	-	4.33%	5.45%
Missing Data	$[871, 1089]$	-	-	5.64%
Perturbed Grid	[980, 1198]	-	-	5.34%
Random Locations	1089	-	-	8.35%
Variable Random Locations	[980, 1198]	-	-	8.28%

5 Discussion

State of the art architectures for operator learning, such as DeepONets [21] and FNOs [16], are significantly restricted in their applicability as they require the number and/or location of input sensors (whether on a Cartesian grid or randomly chosen) to be the same for every training and test sample. This implies that they cannot deal with realistic sources of data and calls into question whether having inputs as vectors of fixed length constitutes operator learning. Thus, it is desirable to design an operator learning framework that can deal with flexible inputs, where number of sensors and their locations vary across samples. We address this need by proposing VIDON (2.6), a novel operator learning framework that allows for such flexible inputs. Moreover, VIDON is designed to be permutation-invariant. We prove that VIDON is universal as well as efficient at approximating operators arising in PDEs. Numerical experiments show that VIDON is comparable in performance to DeepONet, when the latter can be applied. However, the main distinguishing feature of VIDON is its ability to handle inputs with variable number and location of sensors. We observe empirically that for these configurations, VIDON provides robust numerical performance and approximates the underlying operators accurately. At this juncture, we would like to point out some limitations of VIDON, the most prominent being the fact that it is a *heavier* architecture than DeepONets as well as FNOs, requires more storage as well as training time. This seems to be the price to pay for its ability to handle flexible inputs. Moreover, the trunk net of VIDON, as in DeepONet, needs to be learned from data. In future work, one could fix the trunk nets (Fourier basis) or pretrain them [22] to further reduce the training time. Finally, a fixed positional encoding (as in transformers) could replace the learnable encoding, further reducing computational cost.

References

[1] 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.

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

[3] 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.

[4] T. De Ryck, S. Lanthaler, and S. Mishra. On the approximation of functions by tanh neural networks. *Neural Networks*, 2021.

[5] M. Garnelo, D. Rosenbaum, C. Maddison, T. Ramalho, D. Saxton, M. Shanahan, Y. W. Teh, D. J. Rezende, and S. M. A. Eslami. Conditional neural processes. 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 1690–1699. PMLR, 2018.

[6] M. Garnelo, J. Schwarz, D. Rosenbaum, F. Viola, D. J. Rezende, S. M. A. Eslami, and Y. W. Teh. Neural processes. *CoRR*, abs/1807.01622, 2018.

[7] D. Gottlieb and S. A. Orszag. *Numerical analysis of spectral methods: theory and applications*, volume 26. Siam, 1977.

[8] P. Grohs, F. Hornung, A. Jentzen, and P. Von Wurstemberger. A proof that artificial neural networks overcome the curse of dimensionality in the numerical approximation of Black-Scholes partial differential equations. *arXiv preprint arXiv:1809.02362*, 2018.

[9] I. Higgins. Generalizing universal function approximators. *Nature Machine Intelligence*, 3:192–193, 2021.

[10] H. Kim, A. Mnih, J. Schwarz, M. Garnelo, A. Eslami, D. Rosenbaum, O. Vinyals, and Y. W. Teh. Attentive neural processes. In *International Conference on Learning Representations*, 2019.

[11] G. Kissas, J. Seidman, L. Ferreira Guilhoto, V. M. Preciado, G. J. Pappas, and P. Perdikaris. Learning operators with coupled attention. *arXiv preprint arXiv:2202.01032*, 2022.

[12] N. Kovachki, S. Lanthaler, and S. Mishra. On universal approximation and error bounds for Fourier Neural Operators. *arXiv preprint arXiv:2107.07562*, 2021.

[13] N. Kovachki, Z. Li, B. Liu, K. Azizzadensheli, K. Bhattacharya, A. Stuart, and A. Anandkumar. Neural operator: Learning maps between function spaces. *arXiv preprint arXiv:2108.08481v3*, 2021.

[14] 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.

[15] J. Lee, Y. Lee, J. Kim, A. Kosiorek, S. Choi, and Y. W. Teh. Set transformer: A framework for attention-based permutation-invariant neural networks. In *International Conference on Machine Learning*, pages 3744–3753. PMLR, 2019.

[16] Z. Li, N. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. Stuart, and A. Anandkumar. Fourier neural operator for parametric partial differential equations, 2020.

[17] 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.

[18] 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.

[19] 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.

[20] C. Lin, Z. Li, L. Lu, S. Cai, M. Maxey, and G. E. Karniadakis. Operator learning for predicting multiscale bubble growth dynamics. *The Journal of Chemical Physics*, 154(10):104118, 2021.

[21] L. Lu, P. Jin, and G. E. Karniadakis. DeepONet: Learning nonlinear operators for identifying differential equations based on the universal approximation theorem of operators. *arXiv preprint arXiv:1910.03193*, 2019.

[22] L. Lu, X. Meng, S. Cai, Z. Mao, G. Goswami, Z. Zhang, and G. e. Karniadakis. A comprehensive and fair comparison of two neural operators (with practical extensions) based on fair data. *arXiv preprint arXiv:2111.05512v1*, 2021.

[23] Z. Mao, L. Lu, O. Marxen, T. A. Zaki, and G. E. Karniadakis. Deepm&mnet for hypersonics: Predicting the coupled flow and finite-rate chemistry behind a normal shock using neural-network approximation of operators. *Journal of Computational Physics*, 447:110698, 2021.

[24] 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.

[25] T. Tang and J. Yang. Implicit-explicit scheme for the allen-cahn equation preserves the maximum principle. *Journal of Computational Mathematics*, pages 451–461, 2016.

[26] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[27] E. Wagstaff, F. Fuchs, M. Engelcke, I. Posner, and M. A. Osborne. On the limitations of representing functions on sets. In *International Conference on Machine Learning*, pages 6487–6494. PMLR, 2019.

[28] D. Yarotsky. Error bounds for approximations with deep relu networks. *Neural Networks*, 94:103–114, 2017.

[29] M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Poczos, R. R. Salakhutdinov, and A. J. Smola. Deep sets. *Advances in neural information processing systems*, 30, 2017.

A Notation and preliminaries

We introduce notation and preliminary results regarding Sobolev spaces and the Fourier basis, as well as some auxiliary lemmas.

A.1 Glossary of used notation

Table 4: Glossary of used notation.

Symbol	Description	Section
d	spatial dimension of domain	
D, U	general d-dimensional spatial domain	2
$\mathcal{X}$	input function space of the operator $\mathcal{G}$; $\mathcal{X} \subset L^2(D; \mathbb{R}^{d_v})$	2
$\mathcal{Y}$	output function space of the operator $\mathcal{G}$; $\mathcal{Y} \subset L^2(U; \mathbb{R}^{d_u})$	2
$\mathcal{G}$	operator of interest, $\mathcal{G} : \mathcal{X} \to \mathcal{Y}$	2
$\mathfrak{X}^{\leq M}$	the set of subsets of a set $\mathfrak{X}$ containing at most $M \in \mathbb{N}$ elements	2
$\mathfrak{X}^{\mathcal{F}}$	the set of all finite subsets of a set $\mathfrak{X}$	2
$\mathcal{E}$	encoder that maps a function $f : D \to \mathfrak{X}$ to $\mathfrak{X}^{\mathcal{F}}$, cf. (2.2)	2
$\mathcal{E}_m$	encoder that maps a function $f : D \to \mathfrak{X}$ to $\mathfrak{X}^m$, cf. (2.2)	2
$\mathcal{N}$	Variable-Input Deep Operator Network (VIDON); $\mathcal{N} : \mathcal{E}(\mathcal{X}) \to \mathcal{Y}$	2
$\mathbb{T}^d$	periodic torus, identified with $[0, 2\pi)^d$	3
H^s	Sobolev space of smoothness s, with norm $\|\cdot\|_{H^s}$	A.2
$\dot{H}^s$	Sobolev space with zero mean, with norm $\|\cdot\|_{\dot{H}^s}$	A.2
$W^{k,p}$	Sobolev space of smoothness k, with norm $\|\cdot\|_{W^{k,p}}$	A.2
C^k	space of k times continuously differentiable functions	A.2
L^2_N	$L^2_N \subset L^2$ trigonometric polynomials of degree $\leq N$	A.3

A.2 Sobolev spaces

Let $d \in \mathbb{N}$, $k \in \mathbb{N}_0$, $1 \leq p \leq \infty$ and let $\Omega \subseteq \mathbb{R}^d$ be open. For a function $f : \Omega \to \mathbb{R}$ and a (multi-)index $\alpha \in \mathbb{N}_0^d$ we denote by

$$D^\alpha f = \frac{\partial^{|\alpha|} f}{\partial x_1^{\alpha_1} \cdots \partial x_d^{\alpha_d}} \tag{A.1}$$

the classical or distributional (i.e. weak) derivative of f. We denote by $L^p(\Omega)$ the usual Lebesgue space and for we define the Sobolev space $W^{k,p}(\Omega)$ as

$$W^{k,p}(\Omega) = \{f \in L^p(\Omega) : D^\alpha f \in L^p(\Omega) \text{ for all } \alpha \in \mathbb{N}_0^d \text{ with } |\alpha| \leq k\}. \tag{A.2}$$

For $p < \infty$, we define the following seminorms on $W^{k,p}(\Omega)$,

$$|f|_{W^{m,p}(\Omega)} = \left(\sum_{|\alpha|=m} \|D^\alpha f\|^p_{L^p(\Omega)} \right)^{1/p} \qquad \text{for } m = 0, \ldots, k, \tag{A.3}$$

and for $p = \infty$ we define

$$|f|_{W^{m,\infty}(\Omega)} = \max_{|\alpha|=m} \|D^\alpha f\|_{L^\infty(\Omega)} \qquad \text{for } m = 0, \ldots, k. \tag{A.4}$$

Based on these seminorms, we can define the following norm for $p < \infty$,

$$\|f\|_{W^{k,p}(\Omega)} = \left(\sum_{m=0}^{k} |f|^p_{W^{m,p}(\Omega)} \right)^{1/p}, \tag{A.5}$$

and for $p = \infty$ we define the norm

$$\|f\|_{W^{k,\infty}(\Omega)} = \max_{0 \leq m \leq k} |f|_{W^{m,\infty}(\Omega)}. \tag{A.6}$$

The space $W^{k,p}(\Omega)$ equipped with the norm $\|\cdot\|_{W^{k,p}(\Omega)}$ is a Banach space.

We denote by $C^k(\Omega)$ the space of functions that are k times continuously differentiable and equip this space with the norm $\|f\|_{C^k(\Omega)} = \|f\|_{W^{k,\infty}(\Omega)}$.

Lemma A.1 (Continuous Sobolev embedding)**.** *Let $d, \ell \in \mathbb{N}$ and let $k \geq d/2 + \ell$. Then there exists a constant $C > 0$ such that for any $f \in H^k(\mathbb{T}^d)$ it holds that*

$$\|f\|_{C^\ell(\mathbb{T}^d)} \leq C\|f\|_{H^k(\mathbb{T}^d)}. \tag{A.7}$$

A.3 Notation and results for Fourier series

Using the notation from [14], we introduce the following "standard" real Fourier basis $\{\mathbf{e}_\kappa\}_{\kappa\in\mathbb{Z}^d}$ in d dimensions. For $\kappa = (\kappa_1, \ldots, \kappa_d) \in \mathbb{Z}^d$, we let $\sigma(\kappa)$ be the sign of the first non-zero component of κ and we define

$$\mathbf{e}_\kappa := C_\kappa \begin{cases} 1, & \sigma(\kappa) = 0, \\ \cos\big(\langle \kappa, x\rangle\big), & \sigma(\kappa) = -1, \\ \sin\big(\langle \kappa, x\rangle\big), & \sigma(\kappa) = 1, \end{cases} \tag{A.8}$$

where the factor $C_k > 0$ ensures that $\mathbf{e}_\kappa$ is properly normalized, i.e. that $\|\mathbf{e}_\kappa\|_{L^2(\mathbb{T}^d)} = 1$. Next, let $\mathsf{\kappa} : \mathbb{N} \to \mathbb{Z}^d$ be a fixed enumeration of $\mathbb{Z}^d$, with the property that $j \mapsto |\mathsf{\kappa}(j)|_\infty$ is monotonically increasing, i.e. such that $j \leq j'$ implies that $|\mathsf{\kappa}(j)|_\infty \leq |\mathsf{\kappa}(j')|_\infty$. This will allow us to introduced an $\mathbb{N}$-indexed version of the Fourier basis,

$$\mathbf{e}_j(x) := \mathbf{e}_{\mathsf{\kappa}(j)}(x), \quad (j \in \mathbb{N}). \tag{A.9}$$

For $N \in \mathbb{N}$, we define the following two sets

$$\mathcal{J}_N = \{0, \ldots, 2N+1\}^d, \qquad \mathcal{K}_N = \{-N, \ldots, N\}^d, \tag{A.10}$$

following notation from [12]. Furthermore, we denote by $L^2_N(\mathbb{T}^d)$ the space of trigonometric polynomials of degree at most N and we define the following L^2-orthogonal projection on $L^2_N(\mathbb{T}^d)$,

$$P_N : L^2(\mathbb{T}^d) \to L^2_N(\mathbb{T}^d) : \sum_{k\in\mathbb{Z}^d} c_k \mathbf{e}_k \mapsto \sum_{k\in\mathcal{K}_N} c_k \mathbf{e}_k. \tag{A.11}$$

In particular, it holds that if $u \in H^s(\mathbb{T}^d)$ for $s \geq 0$ then there exists a constant $C = C(s,d) > 0$ such that for any $0 \leq \ell \leq s$ it holds that,

$$\|u - P_N u\|_{H^\ell(\mathbb{T}^d)} \leq C N^{\ell - s} \|u\|_{H^s(\mathbb{T}^d)}. \tag{A.12}$$

Similarly, one can define $\dot{P}_N : L^2(\mathbb{T}^d) \to \dot{L}^2_N(\mathbb{T}^d) : u \mapsto P_N u - \fint_{\mathbb{T}^d} u(x)dx$.

Finally, we denote by $\mathcal{I}_N : C(\mathbb{T}^d) \to L^2_N(\mathbb{T}^d)$ the pseudo-spectral projection onto $L^2_N(\mathbb{T}^d)$ i.e., $\mathcal{I}_N u$ is defined as the unique trigonometric polynomial in $L^2_N(\mathbb{T}^d)$ such that $\mathcal{I}_N u(x_j) = u(x_j)$ for all $j \in \mathcal{J}_N$. One can verify that $\mathcal{I}_N u$ is of the form,

$$\mathcal{I}_N u(x) = \frac{1}{|\mathcal{K}_N|} \sum_{k\in\mathcal{K}_N} \sum_{j\in\mathcal{J}_N} a_{k,j} u(x_j) \mathbf{e}_k(x) \tag{A.13}$$

for fixed $a_{k,j}$. For $s, k \in \mathbb{N}_0$ with $s > d/2$ and $s \geq k$, and $u \in C(\mathbb{T}^d) \cap H^s(\mathbb{T}^d)$ it holds that [12]

$$\|u - \mathcal{I}_N u\|_{H^k(\mathbb{T}^d)} \leq C(s,d) N^{-(s-k)} \|u\|_{H^s(\mathbb{T}^d)}, \tag{A.14}$$

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

A.4 Auxiliary results

Lemma A.2. *Let $p \in [2, \infty)$, $d, m \in \mathbb{N}$, let $(\Omega, \mathcal{F}, \mathcal{P})$ be a probability space, and let $X_i : \Omega \to \mathbb{R}^d, i \in \{1, \ldots, m\}$, be i.i.d. random variables with $\mathbb{E}\left[\|X_1\|\right] < \infty$. Then it holds that*

$$\left(\mathbb{E}\left[\left\|\mathbb{E}\left[X_1\right] - \frac{1}{m}\sum_{i=1}^m X_i\right\|^p\right]\right)^{1/p} \leq 2\sqrt{\frac{p-1}{m}} \left(\mathbb{E}\left[\left\|\mathbb{E}\left[X_1\right] - X_1\right\|^p\right]\right)^{1/p}. \tag{A.15}$$

Proof. This result is [8, Corollary 2.5]. □

Lemma A.3. *Let $p \in [2, \infty)$, $q, m \in \mathbb{N}$, let $(\Omega, \mathcal{F}, \mathcal{P})$ and $(\mathcal{D}, \mathcal{A}, \mu)$ be probability spaces, and let for every $q \in \mathcal{D}$ the maps $X_i^q : \Omega \to \mathbb{R}, i \in \{1, \dots, m\}$, be i.i.d. random variables with $\mathbb{E}\left[\left|X_1^q\right|\right] < \infty$. Then it holds that*

$$\mathbb{E}\left[\left(\int_{\mathcal{D}} \left|\mathbb{E}\left[X_1^q\right] - \frac{1}{m}\sum_{i=1}^{m} X_i^q\right|^p \mu(dq)\right)^{1/p}\right] \leq 2\sqrt{\frac{p-1}{m}}\left(\int_{\mathcal{D}} \mathbb{E}\left[\left|\mathbb{E}\left[X_1^q\right] - X_1^q\right|^p\right]\mu(dq)\right)^{1/p}. \tag{A.16}$$

Proof. The proof involves Hölder's inequality, Fubini's theorem and Lemma A.2. The calculation is as in [8, eq. (226)]. □

Lemma A.4. *Let $\varepsilon > 0$, let $(\Omega, \mathcal{F}, \mathcal{P})$ be a probability space, and let $X : \Omega \to \mathbb{R}$ be a random variable that satisfies $\mathbb{E}\left[|X|\right] \leq \varepsilon$. Then it holds that $\mathbb{P}(|X| \leq \varepsilon) > 0$.*

Proof. This result is [8, Proposition 3.3]. □

Lemma A.5. *Let $s, d, p \in \mathbb{N}$. For any $\varepsilon > 0$, there exists a neural network $\boldsymbol{\tau} : \mathbb{R}^d \to \mathbb{R}^p$ with 2 hidden layers of width $\mathcal{O}(p^{\frac{d+1}{d}} + ps\ln(ps\varepsilon^{-1}))$ and such that*

$$p^{3/2} \max_{j=1,\dots,p} \|\tau_j - \mathbf{e}_j\|_{C^s([0,2\pi]^d)} \leq \varepsilon, \tag{A.17}$$

where $\mathbf{e}_1, \dots, \mathbf{e}_p$ denote the first p elements of the Fourier basis, as in Appendix A.3. For fixed s, the number of non-zero weights and biases grows as $\mathcal{O}(p^{\frac{d+2}{d}})$.

Proof. We note that each element in the (real) trigonometric basis $\mathbf{e}_1, \dots, \mathbf{e}_p$ can be expressed in the form

$$\mathbf{e}_j(x) = \cos(\kappa \cdot x), \quad \text{or} \quad \mathbf{e}_j(x) = \sin(\kappa \cdot x), \tag{A.18}$$

for $\kappa = \kappa(j) \in \mathbb{Z}^d$ with $|\kappa|_\infty \leq N$, where N is chosen as the smallest natural number such that $p \leq (2N+1)^d$. We focus only focus on the first form, as the proof for the second form is entirely similar. Define $f : [0, 2\pi]^d \to \mathbb{R} : x \mapsto \kappa \cdot x$ and $g : [-2\pi dN, 2\pi dN] \to \mathbb{R} : x \mapsto \cos(x)$. As $f([0, 2\pi]^d) \subset [-2\pi dN, 2\pi dN]$, the composition $g \circ f$ is well-defined and one can see that it coincides with a trigonometric basis function $\mathbf{e}_j$. Moreover, the linear map f is a trivial neural network without hidden layers. Approximating $\mathbf{e}_j$ by a neural network τ_j therefore boils down to approximating g by a suitable neural network.

From [4, Theorem 5.1] it follows that the function g there exists an independent constant $R > 0$ such that for large enough $t \in \mathbb{N}$ there is a tanh neural network $\widehat{g}_t$ with two hidden layers and $\mathcal{O}(t + N)$ neurons such that

$$\|g - \widehat{g}_t\|_{C^s([-2\pi dN, 2\pi dN])} \leq 4(8(s+1)^3 R)^s \exp(t - s). \tag{A.19}$$

This can be proven from [4, eq. (74)] by setting $\delta \leftarrow \frac{1}{3}$, $k \leftarrow s$, $s \leftarrow t$, $N \leftarrow 2$ and using $\|g\|_{C^s} = 1$ and Stirling's approximation to obtain

$$\frac{1}{(t-s)!}\left(\frac{3}{2\cdot 2}\right)^{t-s} \leq \frac{1}{\sqrt{2\pi(t-s)}}\left(\frac{e}{t-s}\right)^{t-s} \leq \exp(s-t) \quad \text{for } t > s + e^2. \tag{A.20}$$

Setting $t = \mathcal{O}(\ln(\eta^{-1}) + s\ln(s))$ then gives a neural network $\widehat{g}_t$ with $\|g - \widehat{g}_t\|_{C^s} < \eta$. Next, it follows from [4, Lemma A.7] that

$$\begin{aligned} \|g \circ f - \widehat{g}_t \circ f\|_{C^s([0,2\pi]^d)} &\leq 16(e^2 s^4 d^2)^s \|g - \widehat{g}_t\|_{C^s([-2\pi dN, 2\pi dN])} \|f\|^s_{C^s([0,2\pi]^d)} \\ &\leq 16(e^2 s^4 d^2)^s \eta (2\pi dN)^s. \end{aligned} \tag{A.21}$$

From this follows that we can obtain the desired accuracy (A.17) if we set $\tau_j = \widehat{g}_{t(\eta)} \circ f$ with

$$\eta = \frac{\varepsilon p^{-3/2}}{16(2\pi N d^3 e^2 s^4)^s}, \tag{A.22}$$

which amounts to $t = \mathcal{O}(s \ln(sN\varepsilon^{-1}))$. As a consequence, the tanh neural network τ_j has two hidden layers with $\mathcal{O}(s \ln(sN\varepsilon^{-1}) + N)$ neurons and therefore, by recalling that $p \sim N^d$, the combined network $\boldsymbol{\tau}$ has two hidden layers with

$$\mathcal{O}(p(s \ln\Big(sN\varepsilon^{-1}\Big) + N)) = \mathcal{O}(ps \ln\Big(ps\varepsilon^{-1}\Big) + p^{\frac{d+1}{d}}) \tag{A.23}$$

neurons. For fixed s, the number of non-zero weights and biases grows as $\mathcal{O}(2 \cdot p \cdot N^2) = \mathcal{O}(p^{\frac{d+2}{d}})$. □

Lemma A.6. *Let $d, \ell, K, N \in \mathbb{N}$, $v \in H^\ell(\mathbb{T}^d)$, let $(\Omega, \mathcal{F}, P)$ be a probability space and let $X_1, X_2, \ldots : \Omega \to D$ be iid random variables that are uniformly distributed on $\mathbb{T}^d$. Define $V_{K,N} : \Omega \to C^\infty(D)$ by,*

$$V_{K,N}(\omega)(x) = \sum_{|k|_\infty \leq K} \widehat{c}_{k,N}(\omega)\mathbf{e}_k(x), \quad \textit{where} \quad \widehat{c}_{k,N}(\omega) = \frac{|\mathbb{T}^d|}{N} \sum_{n=1}^{N} v(X_n(\omega)) \cdot \mathbf{e}_k(X_n(\omega)). \tag{A.24}$$

Then there exists a constant $C(\ell, d) > 0$ such that for all $s \in \mathbb{N}_0$ with $s \leq \ell$ it holds that,

$$\mathbb{E}\left[\left\|v - V_{K,N}\right\|_{H^s(\mathbb{T}^d)}\right] \leq C(\ell, d)\|v\|_{H^\ell(\mathbb{T}^d)}(K^{s-\ell} + K^{s+d}N^{-1/2}). \tag{A.25}$$

Proof. It holds that v can be written as a Fourier series, $v(x) = \sum_{k \in \mathbb{Z}^d} c_k \mathbf{e}_k(x)$. We can then define

$$v_K(x) = \sum_{|k|_\infty \leq K} c_k \mathbf{e}_k(x) \quad \text{where} \quad c_k = \int_{\mathbb{T}^d} v(x)\mathbf{e}_k(x)dx. \tag{A.26}$$

Using (A.12), we find that,

$$\|v - v_K\|_{H^s(\mathbb{T}^d)} \leq C(\ell, d)\|v\|_{H^\ell(\mathbb{T}^d)}K^{-(\ell-s)}. \tag{A.27}$$

By Lemma A.2 it holds that,

$$\left(\mathbb{E}\left[\left|c_k - \widehat{c}_{k,N}\right|^2\right]\right)^{1/2} \leq \frac{2\|v\|_{L^2(\mathbb{T}^d)}}{\sqrt{N}}. \tag{A.28}$$

Consequently, we find that

$$\begin{aligned}
\mathbb{E}\left[\left\|v_K - V_{K,N}\right\|_{H^s(\mathbb{T}^d)}\right] &= \mathbb{E}\left[\left(\sum_{\alpha \in \mathbb{N}_0^d, \|\alpha\|_1 \leq s} \int_{\mathbb{T}^d} (D^\alpha v_K(x) - D^\alpha V_{K,N}(x))^2 dx\right)^{1/2}\right] \\
&\leq 2(ed)^{s/2}K^s\left|T^d\right|^{1/2} \sum_{|k|_\infty \leq K} \mathbb{E}\left[\left((c_k - \widehat{c}_{k,N})^2\right)^{1/2}\right] \\
&\leq 2(ed)^{s/2}K^s\left|T^d\right|^{1/2} \sum_{|k|_\infty \leq K} \left(\mathbb{E}\left[(c_k - \widehat{c}_{k,N})^2\right]\right)^{1/2} \\
&\leq 4(ed)^{s/2}K^s\left|T^d\right|^{1/2}(2K+1)^d\|v\|_{L^2(\mathbb{T}^d)}N^{-1/2}.
\end{aligned} \tag{A.29}$$

In the first inequality, we used that the number of terms in the sum over α is bounded by $2(ed)^s$ [4, Lemma 2.1], the triangle inequality for the sum over k and the fact that $\|D^\alpha \mathbf{e}_k\|_{C^0} \leq K^s$ for all k (both in the definition of v_K and $V_{K,N}$). Jensen's inequality proves the second inequality. The third inequality follows from (A.28).

As a result, we find that there exists a constant $C(\ell, d) > 0$ such that

$$\mathbb{E}\left[\left\|v - V_{K,N}\right\|_{H^s(\mathbb{T}^d)}\right] \leq C(\ell, d)\|v\|_{H^\ell(\mathbb{T}^d)}(K^{s-\ell} + K^{s+d}N^{-1/2}). \tag{A.30}$$

□

B Proofs of Section 3

Remark B.1. *We will quantify the size of the branch net using the following estimates,*

$$\begin{aligned} \operatorname{depth}(\boldsymbol{\beta}) &\lesssim \operatorname{depth}(\Psi) + \operatorname{depth}(\omega) + \max_{\ell} \operatorname{depth}(\widetilde{\nu}^{(\ell)}) + \operatorname{depth}(\Phi), \\ \operatorname{width}(\boldsymbol{\beta}) &\lesssim \operatorname{width}(\Psi) + \operatorname{width}(\omega) + \textstyle\sum_{\ell=1}^{H} \operatorname{width}(\widetilde{\nu}^{(\ell)}) + \operatorname{width}(\Phi), \\ \operatorname{size}(\boldsymbol{\beta}) &\lesssim \operatorname{size}(\Psi) + \operatorname{size}(\omega) + \textstyle\sum_{\ell=1}^{H} \operatorname{size}(\widetilde{\nu}^{(\ell)}) + \operatorname{size}(\Phi), \end{aligned} \tag{B.1}$$

where we define the size of a neural network as its number of non-zero weights and biases. The size of the Variable-Input Deep Operator Network can be estimated by $\operatorname{size}(\mathcal{N}) \leq \operatorname{size}(\boldsymbol{\beta}) + \operatorname{size}(\boldsymbol{\tau})$. *Note that these sizes reflect the total number of unique parameters, rather than the computational complexity of evaluating a Variable-Input Deep Operator Network, as the latter will also depend on the number of sensors* m.

B.1 Proof of Theorem 3.1 (Universal approximation)

Proof. Let $x_1, \dots x_M$ be iid random variables on D, let $\mathcal{E}_m : \mathcal{X} \to \mathbb{R}^m : u_0 \mapsto \{(x_j, u_0(x_j))\}_{j=1}^m$, $1 \leq m \leq M$, be encoders and let $\mathcal{D}_m : \mathbb{R}^m \to \mathcal{X}$ be the corresponding decoders as defined in [14, (3.38)]. We can then define another encoder $\mathcal{E} : \mathcal{X} \times \{1, \dots, M\} \to \mathbb{R}^{\leq M} : (u_0, m) \to \mathcal{E}_m(u_0)$ and the corresponding decoder $\mathcal{D} : \mathbb{R}^{\leq M} \to \mathcal{X} : X \to D_{|X|}(X)$. The continuity of $\mathcal{D}$ (where we interpret the continuity of a function of sets as in [27, Section A.2]) follows from its definition in [14, (3.38)] and the continuity of the Moore-Penrose inverse on the set of matrices of fixed size with full rank.

We will prove that there exists a Variable-Input Deep Operator Network of the form $\mathcal{N} = \mathcal{R} \circ \mathcal{A}$, where $\mathcal{A} : \mathbb{R}^{\leq M} \to \mathbb{R}^p$ is an approximation operator and $\mathcal{R} : \mathbb{R}^p \to C(U)$ is a reconstruction operator, both of which will be exactly defined later on. We write $\mathcal{P} : C(U) \to \mathbb{R}^p$ for the projection operator corresponding to $\mathcal{R}$. Such a decomposition has been proposed and studied in [14]. Moreover, it holds that $\mathcal{E} \circ \mathcal{D} = \mathrm{Id}$, $\mathcal{D} \circ \mathcal{E} \approx \mathrm{Id}$, $\mathcal{P} \circ \mathcal{R} = \mathrm{Id}$ and $\mathcal{R} \circ \mathcal{P} \approx \mathrm{Id}$. In particular, it has been proven that the DeepONet error decomposes in the following way [14, Lemma 3.4],

$$\left\| \mathcal{G}(u_0) - \mathcal{N}(\mathcal{E}(u_0)) \right\|_{L^2(\mu)} \leq C \operatorname{Lip}_\alpha(\mathcal{G}) \operatorname{Lip}(\mathcal{R} \circ \mathcal{P}) (\widehat{\mathscr{E}}_{\mathcal{E}})^\alpha + \operatorname{Lip}_\alpha(\mathcal{R}) \widehat{\mathscr{E}}_{\mathcal{A}} + \widehat{\mathscr{E}}_{\mathcal{R}}, \tag{B.2}$$

where $\widehat{\mathscr{E}}_{\mathcal{E}}$, $\widehat{\mathscr{E}}_{\mathcal{A}}$, $\widehat{\mathscr{E}}_{\mathcal{R}}$ are the errors related to $\mathcal{E}$, $\mathcal{A}$, $\mathcal{R}$, respectively (see [14, Section 3.2.1] for exact definitions). We will now prove an upper bound for each separate term.

First, we use [14, Theorem 3.7] to conclude that $\widehat{\mathscr{E}}_{\mathcal{E}}$ can be bounded in terms of the eigenvalues of the covariance operator,

$$\widehat{\mathscr{E}}_{\mathcal{E}} \leq C \left(\sum_{j > m/C \log(m)} \lambda_j \right)^{\alpha/2} =: \eta(m) \qquad \text{(with probability 1)}. \tag{B.3}$$

Next, we choose $\mathcal{R}$ to be a Fourier-based reconstruction operator,

$$\mathcal{R} : \mathbb{R}^p \to C(U) : (\alpha_1, \dots, \alpha_p) \mapsto \sum_{j=1}^{p} \alpha_j \widehat{\mathbf{e}}_j, \tag{B.4}$$

where the $\widehat{\mathbf{e}}_j$ are neural network approximations of the Fourier basis $\{\mathbf{e}_j\}_j$ (Appendix A.3 and Lemma A.5). Using [14, Theorem 3.5], we find that $\widehat{\mathscr{E}}_{\mathcal{R}} \leq C p^{-s/d}$ and $\operatorname{Lip}(\mathcal{R} \circ \mathcal{P}), \operatorname{Lip}_\alpha(\mathcal{R}) \leq C$. We thus can rewrite (B.2) as,

$$\left\| \mathcal{G}(u_0) - \mathcal{N}(\mathcal{E}(u_0)) \right\|_{L^2(\mu)} \leq C(\eta(m) + \widehat{\mathscr{E}}_{\mathcal{A}} + p^{-s/d}) \qquad \text{(with probability 1)}, \tag{B.5}$$

where η is monotonically decreasing in m and converging to 0, as in (B.3).

It remains to bound the approximation error $\widehat{\mathscr{E}}_{\mathcal{A}}$, which quantifies how well $\mathcal{A}$ approximates $\mathcal{P} \circ \mathcal{G} \circ \mathcal{D} : \mathbb{R}^{\leq M} \to \mathbb{R}^p$. By definition and because of the choice of random sensors, $\mathcal{P} \circ \mathcal{G} \circ \mathcal{D}$ is a continuous permutation-invariant function. It then follows from [27, Theorem 4.4] (which builds upon the

work of [29]) that $\mathcal{P} \circ \mathcal{G} \circ \mathcal{D}$ is continuously sum-decomposable via $\mathbb{R}^M$, meaning that there exist continuous functions $\varphi : \mathbb{R} \to \mathbb{R}^M$ and $\rho : \mathbb{R}^M \to \mathbb{R}^p$ such that for every $X \in \mathbb{R}^{\leq M}$ it holds that $(\mathcal{P} \circ \mathcal{G} \circ \mathcal{D})(X) = \rho\left(\sum_{x \in X} \varphi(x)\right)$. By the universal approximation property of neural networks, for arbitrary $\delta, \varepsilon > 0$ there exist networks $\widehat{\varphi}_\varepsilon$ and $\widehat{\rho}_\delta$ such that

$$\|\varphi - \widehat{\varphi}_\varepsilon\|_{C^0} < \varepsilon \qquad \text{and} \qquad \|\rho - \widehat{\rho}_\delta\|_{C^0} < \delta. \tag{B.6}$$

Moreover, $\widehat{\rho}_\delta$ is Lipschitz continuous with Lipschitz constant L_δ. We can then define $\mathcal{A}(X) = \widehat{\rho}_\delta\left(\sum_{x \in X} \widehat{\varphi}_\varepsilon(x)\right)$ for every $X \in \mathbb{R}^{\leq m}$. Using the triangle inequality then gives that for any $X \in \mathbb{R}^{\leq M}$ and $y := \sum_{x \in X} \varphi(x)$,

$$\begin{aligned} \left|\mathcal{A}(X) - (\mathcal{P} \circ \mathcal{G} \circ \mathcal{D})(X)\right| &\leq C_\delta \sum_{x \in X} \left|\widehat{\varphi}_\varepsilon(x) - \varphi(x)\right| + \left|\widehat{\rho}_\delta(y) - \rho(y)\right| \\ &\leq C_\delta M \varepsilon + \delta. \end{aligned} \tag{B.7}$$

Combining with (B.5) then gives us,

$$\left\|\mathcal{G}(u_0) - \mathcal{N}(\mathcal{E}(u_0))\right\|_{L^2(\mu)} \leq C(\eta(m) + C_\delta M \varepsilon + \delta + p^{-s/d}) \qquad \text{(with probability 1).} \tag{B.8}$$

Setting $\varepsilon = \delta/(C_\delta M)$ and $\delta = p^{-s/d}$ then gives us the error estimate from the statement.

We conclude the proof by observing that $\mathcal{N} = \mathcal{R} \circ \mathcal{A}$ indeed fits in the proposed architecture of Section 2. For the branch net we set $d_{enc} := d_v$, $\Psi_c := 0$, $\Psi_v \approx \mathrm{Id}$, $\omega := 1$, $\widetilde{\nu} := \widehat{\varphi}_\varepsilon$, $H := 1$ and $\Phi := \widehat{\rho}_\delta$. The trunk nets are given by $\tau_j := \widehat{\mathrm{e}}_j$ for all j. □

Remark B.2. *If we assume that the function $\rho : \mathbb{R}^m \to \mathbb{R}^p$ from the above proof of Theorem 3.1 is Lipschitz continuous, then it can be proven that one needs a network (roughly) of size $\mathcal{O}(\varepsilon^{-m})$ to approximate ρ to an accuracy $\varepsilon > 0$, using a result in the sense of [28, 4]. Using the function $\eta : \mathbb{N} \to [0, \infty)$ from* (B.3)*, we define its approximate inverse $\xi : (0, \infty) \to \mathbb{N} : \varepsilon \mapsto \inf\{n \in \mathbb{N} \mid \eta(n) < \varepsilon\}$, which quantifies the needed number of sensors to obtain a certain accuracy. As a result, a lower bound for the network size to approximate ρ is given by $\mathcal{O}(\varepsilon^{-\xi(\varepsilon)})$. Depending on the chosen measure, this can grow rapidly. For the Gaussian setting of Corollary 3.2 it holds that $\xi(\varepsilon) \sim \sqrt{\log(1/\varepsilon)}$.*

B.2 Proof of Theorem 3.3 (Darcy flow)

Proof. In the following, we let $r := \ell - d$ and we will use notation from Section A.3. For $K, N \in \mathbb{N}$, one can use Lemma A.6 to define a random variable $a_{K,N}$ for which it holds that $\mathbb{E}\left[\left\|a - a_{K,N}\right\|_{H^d(\mathbb{T}^d)}\right] \leq C(K^{d-\ell} + K^{2d} N^{-1/2})$. Using a Sobolev embedding theorem, we find that,

$$\left\|a - a_{K,N}\right\|_{L^\infty(\mathbb{T}^d)} \leq C\left\|a - a_{K,N}\right\|_{H^d(\mathbb{T}^d)} \leq C(K^{-r} + K^{2d} N^{-1/2}). \tag{B.9}$$

From this follows that $\inf_{x \in \mathbb{T}^d} a_{K,N}(x) \geq \frac{\lambda}{2} > 0$ if K and N are suitably chosen, given that $\inf_{x \in \mathbb{T}^d} a(x) \geq \lambda > 0$. Combining this with [14, Lemma 4.7] then gives for the solution $u_{K,N}$ of the Darcy flow with $a_{K,N} := a_{K,N}(\omega)$ (for a fixed $\omega \in \Omega$) instead of a, that,

$$\left\|u - u_{K,N}\right\|_{L^2(\mathbb{T}^d)} \leq C\left\|a - a_{K,N}\right\|_{L^\infty(\mathbb{T}^d)} \leq C(K^{-r} + K^{2d} N^{-1/2}). \tag{B.10}$$

It also holds that $\mathbb{E}\left[\left\|a_{K,N}\right\|_{H^\ell(\mathbb{T}^d)}\right] \leq \|a\|_{H^\ell(\mathbb{T}^d)} + C K^{\ell+d} N^{-1/2} \leq \lambda^{-1}$ (from Lemma A.6) and hence by letting $N^{1/2} \geq K^{\ell+d}$ we find $\mathbb{E}\left[\left\|a_{K,N}\right\|_{H^\ell(\mathbb{T}^d)}\right] \leq \lambda^{-1}$ where λ might have to be redefined (increased). By [12, Theorem 3.5], for any $M \in \mathbb{N}$ there exists a Ψ-FNO $\mathcal{N}_\Psi$ such that

$$\left\|u_{K,N} - \mathcal{N}_\Psi(a_{K,N})\right\|_{H^1(\mathbb{T}^d)} \leq C M^{-r}, \tag{B.11}$$

where the width of $\mathcal{N}_\Psi$ grows as $\mathcal{O}(M^d)$ and the depth grows as $\mathcal{O}(\log(M))$. A Ψ-FNO is a discrete realizeation of an FNO [12, Definition 11] and maps $\{a_{K,N}(x_j)\}_{j \in \mathcal{J}_{2M}}$ to $\{\mathcal{N}_\Psi(a_{K,N})(x_j)\}_{j \in \mathcal{J}_M}$, then applies a linear mapping $\mathcal{B}_M$ to obtain coefficients $\{\beta_m(a_{K,N})\}_{m \in \mathcal{K}_M}$

such that $\mathcal{N}_\Psi(a_{K,N})(x) = \sum_{j\in\mathcal{K}_M} \beta_m(a_{K,N})\mathbf{e}_j(x)$. It is important to note that $a_{K,N}(x_j)$ can be written as

$$a_{K,N}(\omega)(x_j) = \frac{|\mathbb{T}^d|}{N}\sum_{n=1}^{N}\sum_{|k|_\infty\leq K} a(X_n(\omega))\cdot \mathbf{e}_k(X_n(\omega))\cdot \mathbf{e}_k(x_j), \tag{B.12}$$

which is permutation-invariant with respect to $\{X_n\}_n$. Next, Lemma A.5 guarantees that there exist a tanh neural network with two hidden layers, each with $\mathcal{O}(K^d\ln(\delta^{-1}) + K^{d+1})$ neurons, such that $\|\mathbf{e}_k - \widehat{\mathbf{e}}_k\|_{C^0} \leq \delta$. We then define $\widehat{a_{K,N}}$ by replacing $\mathbf{e}_k$ by $\widehat{\mathbf{e}}_k$ and the first multiplication in (B.12) by a (fixed-size) neural network $\widehat{\times}$ that approximates the multiplication operator in C^0-norm, which can be done to arbitrary accuracy with a fixed size neural network [4]. More specifically, this leads to,

$$\widehat{a_{K,N}}(\omega)(x_j) = \frac{|\mathbb{T}^d|}{N}\sum_{n=1}^{N}\sum_{|k|_\infty\leq K} a(X_n(\omega))\widehat{\times}\,\widehat{\mathbf{e}}_k(X_n(\omega))\cdot \mathbf{e}_k(x_j), \tag{B.13}$$

Applying the Ψ-FNO $\mathcal{N}_\Psi$ and linear mapping $\mathcal{B}_M$ as before then gives rise to the coefficients $\{\beta_m(\widehat{a_{K,N}})\}_{m\in\mathcal{K}_M}$. We then define $\mathcal{N}$ as,

$$\mathcal{N}(\omega)(a)(x) = \sum_{|k|_\infty\leq K} \beta_m(\widehat{a_{K,N}})\widehat{\mathbf{e}}_k(x). \tag{B.14}$$

By comparing (B.12) and (B.13), using the Lipschitz continuity of $\mathcal{N}_\Psi$ and the accuracy of $\widehat{\times}$ and $\mathbf{e}_k$, we find for a fixed a that,

$$\big\|\mathcal{N}(\omega)(a) - \mathcal{N}_\Psi(\omega)(a_{K,N})\big\|_{L^2(\mathbb{T}^d)} \leq CK^d\delta. \tag{B.15}$$

By redefining $\delta \leftarrow K^{-r-d}$ and by setting the number of sensors points as $m \leftarrow N$ and the number of branch nets as $p \leftarrow K^d = M^d$, we find the following total error estimate,

$$\mathbb{E}\left[\big\|\mathcal{N}(a) - \mathcal{G}(a)\big\|_{L^2(\mathbb{T}^d)}\right] \leq C(p^{-r/d} + p^2 m^{-1/2}). \tag{B.16}$$

Finally, we see that $\mathcal{N}$ is indeed a Variable-Input Deep Operator Network by comparing with the architecture in Section 2 with $d_{enc} := d + d_v$, $\Psi \approx \mathrm{Id}$, $H := 1$, $\omega := 1$,

$$\widetilde{\nu}_j((x, a(x))) = |\mathbb{T}^d|\cdot \sum_{|k|_\infty\leq K} \mathbf{e}_k(x_j)\cdot a(x)\,\widehat{\times}\,\widehat{\mathbf{e}}_k(x) \quad \text{for every } j\in\mathcal{J}_M, \tag{B.17}$$

$\Phi := \mathcal{B}_M \circ \mathcal{N}_\Psi$ and $\tau_k := \widehat{\mathbf{e}}_k$. The width and depth of Ψ and ω are $\mathcal{O}(1)$, furthermore it holds that $\mathrm{depth}(\widetilde{\nu}) = 3$ (2 for $\widehat{\mathbf{e}}_k$ and 1 for $\widehat{\times}$), $\mathrm{width}(\widetilde{\nu}) = \mathcal{O}(p\ln(p) + p^{(d+1)/d}) = \mathcal{O}(p^{(d+1)/d})$ and $\mathrm{size}(\widetilde{\nu}) = \mathcal{O}(p^{(d+2)/d})$ (the latter as a result from Lemma A.5). Also, $\mathrm{depth}(\Phi) = \mathcal{O}(\log(p))$ and $\mathrm{width}(\Phi) = \mathcal{O}(p)$ and therefore $\mathrm{size}(\Phi) = \mathcal{O}(p^2\log(p))$. Using (B.1) we find,

$$\mathrm{depth}(\boldsymbol{\beta}) = \mathcal{O}(\log(p)), \quad \mathrm{width}(\boldsymbol{\beta}) = \mathcal{O}(p^{(d+1)/d}) \quad \text{and} \quad \mathrm{size}(\boldsymbol{\beta}) = \mathcal{O}(p^3), \tag{B.18}$$

where we used the upper bound that $p^{(d+2)/d} + p^2\log(p) \lesssim p^3$. For the trunk net we find that $\mathrm{depth}(\boldsymbol{\tau}) = 2$, $\mathrm{width}(\boldsymbol{\tau}) = \mathcal{O}(p^{(d+1)/d})$ and $\mathrm{size}(\boldsymbol{\tau}) = \mathcal{O}(p^{(d+2)/d})$ (as a result from Lemma A.5).

Moreover, if we want to obtain an accuracy of $\varepsilon > 0$, we need to set $p := \varepsilon^{-d/r}$ and $m := \varepsilon^{-2(d+\ell)/r}$. This gives then rise to a total size (B.1) of $\mathrm{size}(\mathcal{N}) = \mathcal{O}(\varepsilon^{-3d/r})$. □

B.3 Proof of Theorem 3.4 (Allen-Cahn)

Proof. For $K, N \in \mathbb{N}$, one can use Lemma A.6 to define a random variable $u_0^{K,N}$, defined by,

$$u_0^{K,N}(\omega)(x) = \sum_{|k|_\infty\leq K} \widehat{c}_{k,N}(\omega)\cdot \mathbf{e}_k(x) \quad \text{where} \quad \widehat{c}_{k,N}(\omega) = \frac{|\mathbb{T}^d|}{N}\sum_{n=1}^{N} u_0(X_n(\omega))\cdot \mathbf{e}_k(X_n(\omega)), \tag{B.19}$$

for which it holds that $\mathbb{E}\left[\left\|u_0 - u_0^{K,N}\right\|_{H^s(\mathbb{T}^d)}\right] \leq C(K^{s-\ell} + K^{s+d}N^{-1/2})$ for any $0 \leq s \leq \ell$. Using a Sobolev embedding theorem we find (for a fixed realization of $u_0^{K,N} = u_0^{K,N}(\omega)$

that $\left\|u_0 - u_0^{K,N}\right\|_{C^0(\mathbb{T}^d)} \leq \left\|u_0 - u_0^{K,N}\right\|_{H^d(\mathbb{T}^d)}$. Lemma A.5 guarantees that there exist a tanh neural network with two hidden layers, each with $\mathcal{O}(K^d \ln(\delta^{-1}) + K^{d+1})$ neurons, such that $\|\mathbf{e}_k - \widehat{\mathbf{e}}_k\|_{C^0} \leq \delta$ and in [4] it is proven that one can approximate the multiplication operator in C^0-norm to accuracy δ with a fixed size neural network $\widehat{\times}$. Using these definitions, we define

$$U_j^0 = \frac{|\mathbb{T}^d|}{N} \sum_{|k|_\infty \leq K} \sum_{n=1}^{N} u_0(X_n(\omega)) \widehat{\times} \widehat{\mathbf{e}}_k(X_n(\omega)) \cdot \mathbf{e}_k(x_j) \quad \text{for } j \in \mathcal{J}_M,\ M \in \mathbb{N}, \tag{B.20}$$

for which it holds that,

$$\mathbb{E}\left[\max_j \left|U_j^0 - u_0(x_j)\right|\right] \leq CK^d(K^{-\ell} + K^d N^{-1/2} + \delta). \tag{B.21}$$

Next, we will approximate all $u(T, x_j)$ for $j \in \mathcal{J}_M$, $M \in \mathbb{N}$. In [25, Theorem 4.1] a finite difference scheme was proposed that takes $U_j^0 \approx u_0(x_j)$ as input and returns an approximation $U_j^n \approx u(T, x_j)$ for $n \in \mathbb{N}$ with T/n being the time step of the scheme. Using the refinement of this result from [14, Theorem 4.14] we find the error estimate

$$\mathbb{E}\left[\max_j \left|U_j^n - u(T, x_j)\right|\right] \leq \exp\left(CT\|u\|_{C^{(2,4)}([0,T]\times\mathbb{T}^d)}\right) \cdot \left(n^{-1} + M^{-2} + \mathbb{E}\left[\max_j \left|U_j^0 - u_0(x_j)\right|\right]\right). \tag{B.22}$$

Following [14, Theorem 4.11] we emulate this finite difference scheme to create a neural network approximation $\widehat{U}_j^n$ of U_j^n. Using the results on function approximation by tanh neural networks from [4] we find that there exists a tanh neural network $\widehat{U}^M$ of width $\mathcal{O}(M^d)$ and depth $\mathcal{O}(n)$ that maps $\{U_j^0\}_j$ to $\{\widehat{U}_j^n\}_j$ for which it holds that,

$$\mathbb{E}\left[\max_j \left|\widehat{U}_j^n - u(T, x_j)\right|\right] \leq C(n^{-1} + M^{-2} + K^d(K^{-\ell} + K^d N^{-1/2} + \delta)), \tag{B.23}$$

where we used (B.22).

Now define v^Z, $Z \in \mathbb{N}$ where Z is a divisor of M, as the trigonometric polynomial interpolation of $u(T)$ at the points $\{x_j\}_{j\in\mathcal{J}_Z} \subset \{x_j\}_{j\in\mathcal{J}_M}$ (see Section A.3 for more information). One can then define the function $\widehat{v}^Z$ by

$$\widehat{v}^Z(x) = \frac{1}{|\mathcal{K}_Z|} \sum_{k\in\mathcal{K}_Z} \sum_{j\in\mathcal{J}_Z} a_{k,j} \widehat{U}_j^M \mathbf{e}_k(x), \tag{B.24}$$

which inspires us to define the Variable-Input Deep Operator Network by

$$\mathcal{N}(u_0)(x) = \frac{1}{|\mathcal{K}_Z|} \sum_{k\in\mathcal{K}_Z} \sum_{j\in\mathcal{J}_Z} a_{k,j} \widehat{U}_j^n \widehat{\mathbf{e}}_k(x), \tag{B.25}$$

where $\{\widehat{\mathbf{e}}_k\}_k$ is a tanh neural network with two hidden layers, each with width $\mathcal{O}(Z^d \ln(\varepsilon^{-1}) + Z^{d+1})$ and size $\mathcal{O}(Z^d \ln(\varepsilon^{-1})^2 + Z^{d+2})$, such that $\|\mathbf{e}_k - \widehat{\mathbf{e}}_k\|_{C^0} \leq \varepsilon$ (Lemma A.5). By comparing (B.24) with (A.14) and combining this with (B.23) we find that,

$$\mathbb{E}\left[\left\|\mathcal{G}(u_0) - \mathcal{N}(u_0)\right\|_{L^2(\mathbb{T}^d)}\right] \lesssim C(Z^{-\ell} + Z^d(n^{-1} + M^{-2} + K^d(K^{-\ell} + K^d N^{-1/2} + \delta) + \varepsilon)). \tag{B.26}$$

The error estimate from the statement then follows by redefining $n \leftarrow Z^{d+\ell}$, $M \leftarrow Z^{(d+\ell)/2}$, $K \leftarrow Z^{(\ell+d)/(\ell-d)}$, $\delta \leftarrow K^{-\ell}$, $\varepsilon \leftarrow Z^{-d-\ell}$ and by setting the number of sensors points as $m \leftarrow N$ and the number of branch nets as $p \leftarrow Z^d$. Indeed, we find that,

$$\mathbb{E}\left[\left\|\mathcal{G}(u_0) - \mathcal{N}(u_0)\right\|_{L^2(\mathbb{T}^d)}\right] \leq C(p^{-\ell/d} + p^{3(\ell+d)/(\ell-d)} m^{-1/2}) \tag{B.27}$$

Finally, we see that $\mathcal{N}$ is indeed a Variable-Input Deep Operator Network by comparing with the architecture in Section 2 with $d_{enc} := d + d_v$, $\Psi \approx \mathrm{Id}$, $H := (M/Z)^d$, $\omega := 1$,

$$\widetilde{\nu}^{(h)}((x, u_0(x))) = |\mathbb{T}^d| \cdot \sum_{|k|_\infty \leq K} \alpha_k^{(h)} \cdot u_0(x) \widehat{\times} \widehat{\mathbf{e}}_k(x), \qquad 1 \leq h \leq H, \tag{B.28}$$

where the $\alpha_k^{(h)}$ are defined in such a way that the output of ν is equal to $\{U_0^j\}_j$ as in (B.20). Finally we set $\Phi := \widehat{U}^M$ and $\tau_k := \widehat{\mathbf{e}}_k$. The width and depth of Ψ and ω are $\mathcal{O}(1)$. It also holds that $\mathrm{depth}(\widetilde{\nu}) = 3$ (2 for $\widehat{\mathbf{e}}_k$ and 1 for $\widehat{\times}$),

$$\begin{aligned} \sum_{h=1}^{H} \mathrm{width}(\widetilde{\nu}^{(h)}) &= \mathcal{O}(H \cdot Z^{d+1}) = \mathcal{O}(M^d Z) = \mathcal{O}(Z^{d(d+\ell)/2+1}) = \mathcal{O}(p^{(d+\ell)/2+1/d}), \\ \sum_{h=1}^{H} \mathrm{size}(\widetilde{\nu}^{(h)}) &= \mathcal{O}(H \cdot 3 \cdot Z^{d+2}) = \mathcal{O}(M^d Z^2) = \mathcal{O}(Z^{d(d+\ell)/2+2}) = \mathcal{O}(p^{(d+\ell)/2+2/d}). \end{aligned} \tag{B.29}$$

Furthermore we find that $\mathrm{depth}(\Phi) = \mathcal{O}(n) = \mathcal{O}(p^{1+\ell/d})$ and $\mathrm{width}(\Phi) = \mathcal{O}(M^d) = \mathcal{O}(p^{(d+\ell)/2})$. Using (B.1) we find,

$$\mathrm{depth}(\boldsymbol{\beta}) = \mathcal{O}(p^{1+\ell/d}), \quad \mathrm{width}(\boldsymbol{\beta}) = \mathcal{O}(p^{(d+\ell)/2+1/d}) \quad \text{and} \quad \mathrm{size}(\boldsymbol{\beta}) = \mathcal{O}(p^{(d+\ell)/2+2/d}). \tag{B.30}$$

For the trunk net we find that $\mathrm{depth}(\boldsymbol{\tau}) = 2$, $\mathrm{width}(\boldsymbol{\tau}) = \mathcal{O}(p^{(d+1)/d})$ and $\mathrm{size}(\boldsymbol{\tau}) = \mathcal{O}(p^{(d+2)/d})$.

Moreover, if we want to obtain an accuracy of $\varepsilon > 0$, we need to set $p := \varepsilon^{-d/\ell}$ and $m := \varepsilon^{-2-6d(\ell+d)/\ell(\ell-d)}$. This gives then rise to a total size (B.1) of $\mathrm{size}(\mathcal{N}) = \mathcal{O}(\varepsilon^{-d(d+\ell)/2\ell-2\ell})$.

□

B.4 Proof of Theorem 3.5 (Navier-Stokes)

Definition B.3. *Let $T > 0$, $\nu \geq 0$, $d \geq 2$, $r \geq d/2 + 2$. We define $\mathcal{V} \subset C([0,T]; H^r) \cap C^1([0,T]; H^{r-2})$ as the set of solutions of the Navier-Stokes equations* (3.7)*, such that $\sup_{u \in \mathcal{V}} \|u\|_{L^2} < \infty$, and*

$$\sup_{u \in \mathcal{V}} \left\{ \|u\|_{C_t(H_x^r)} + \|u\|_{C_t^1(H_x^{r-2})} \right\} < \infty. \tag{B.31}$$

Proof of Theorem 3.5. The proof is rather similar to that of Theorem 3.3. To avoid too many indices, we define $a := u_0$. For $K, N \in \mathbb{N}$, one can use Lemma A.6 to define a random variable $a_{K,N}$ for which it holds that $\mathbb{E}\left[\left\|a - a_{K,N}\right\|_{H^s(\mathbb{T}^d)}\right] \leq C(K^{s-r} + K^{s+d} N^{-1/2})$ for any $0 \leq s \leq r$. By [12, Theorem 3.12], for any $M \in \mathbb{N}$ there exits a Ψ-FNO $\mathcal{N}_\Psi^*$ such that

$$\left\|u - \mathcal{N}_\Psi^*(a)\right\|_{L^2(\mathbb{T}^d)} \leq C M^{-r}, \tag{B.32}$$

where the width of $\mathcal{N}_\Psi^*$ grows as $\mathcal{O}(M^d)$ and the depth grows as $\mathcal{O}(\log(M))$. In a similar way, we can find a Ψ-FNO $\mathcal{N}_\Psi$ for which $\left\|u - \mathcal{N}_\Psi^*(a_{K,N})\right\|_{L^2(\mathbb{T}^d)}$ is small. This can be done by making the small adaptation in the proof in [12] of using $\dot{P}_M \mathcal{I}_{2M} a_{K,N}$ as input for the Ψ-FNO instead of $\dot{P}_M \mathcal{I}_{2M} a$. In [12, Section F.2.5] they use that

$$\left\|(1 - \dot{P}_M \mathcal{I}_{2M}) a\right\|_{L^2(\mathbb{T}^d)} \leq M^{-r} \|a\|_{H^r(\mathbb{T}^d)}. \tag{B.33}$$

If we combine the estimate,

$$\left\|(1 - \dot{P}_M \mathcal{I}_{2M}) a_{K,N}\right\|_{L^2(\mathbb{T}^d)} \leq C M^{-r} \left(\|a\|_{H^r(\mathbb{T}^d)} + \left\|a - a_{K,N}\right\|_{H^r(\mathbb{T}^d)} \right), \tag{B.34}$$

with the properties of $a_{K,N}$ (i.e. Lemma A.6 with $s = 0$) then we find that

$$\mathbb{E}\left[\left\|a - \dot{P}_M \mathcal{I}_{2M} a_{K,N}\right\|_{L^2(\mathbb{T}^d)}\right] \leq C(K^{-r} + K^d N^{-1/2} + M^{-r}(C + K^{r+d} N^{-1/2})). \tag{B.35}$$

By replacing (B.33) with (B.35) in [12, Section F.2.5], we find that there exists a Ψ-FNO $\mathcal{N}_\Psi$ such that

$$\left\|u - \mathcal{N}_\Psi^*(a_{K,N})\right\|_{L^2(\mathbb{T}^d)} \leq C(K^{-r} + K^d N^{-1/2}(1 + K^r M^{-r})). \tag{B.36}$$

The proof can be finished in a similar way to the proof of Theorem 3.3. In particular, we set $m \leftarrow N$, $p \leftarrow K^d$ and $M = K$. The sizes of $\boldsymbol{\beta}$ and $\boldsymbol{\tau}$ are the same in terms of p, m as in the proof of Theorem 3.3. However, if we want to obtain an accuracy of $\varepsilon > 0$, we now need to set $p := \varepsilon^{-d/r}$ and $m := \varepsilon^{-2-2d/r}$. This gives then rise to a total size (B.1) of $\text{size}(\mathcal{N}) = \mathcal{O}(p^{-3d/r})$. □

C Details for Numerical experiments in Section 4

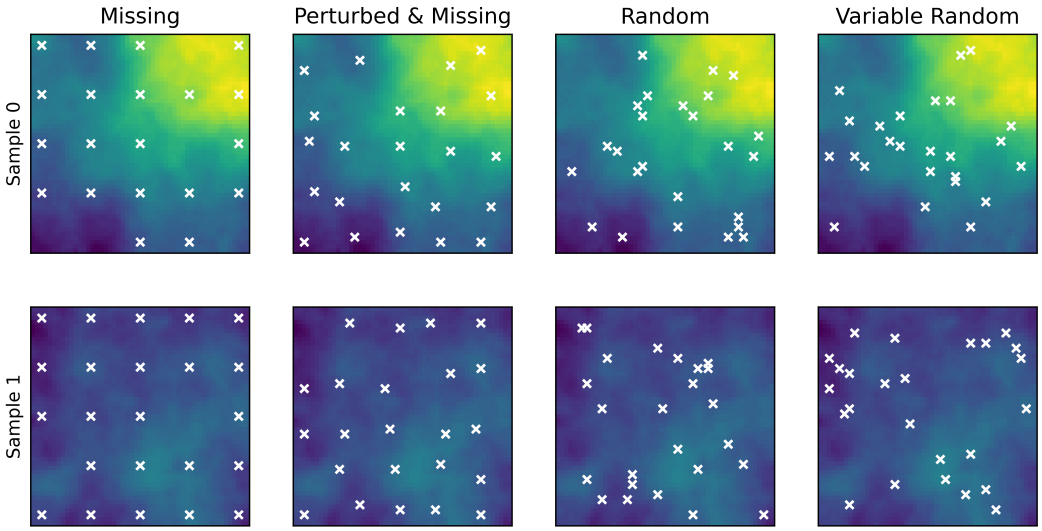


Figure 3: Illustration of sensor locations for configuration where sensors are not on a grid. See Section 4 for the nomenclature. We consider two sample inputs for the Darcy flow test case. See columns 2 and 3 in Figure 1 for illustrations of regular and irregular grids respectively and plot the other sensor configurations here.

C.1 Training and Architecture Details

For each of the three problems we create a training set containing 1000 samples, a validation set containing 32 samples and a test set containing 5000 samples. The datasets for each coordinate configuration (regular grid, irregular grid, etc) use the same initial condition, so all models are trained on the same underlying samples. For training and validation sets the inputs and outputs (where the model is evaluated) use the same sensor coordinates. For example, when sensor points are drawn at random then the output is only available at the sensor locations which were also used in the input. This is intended to simulate physical measurements where data will only ever be available at the given sensor locations. For the test sets the input is given at the respective sensor points but the output is evaluated on a fine grid. This is done to get an accurate estimate of how well the models learn the true solutions and to see how well the models generalize to arbitrary points in the domain. The corresponding grid sizes for (training and) testing are given in table 5.

During the training process, optimization is performed using the ADAM optimizer with the mean squared error loss. In every epoch the relative L^2 error is monitored on the (very small) validation set. The model that achieves the lowest relative L^2 error on the validation set (during the training process) is saved and selected for testing. We obtain the model hyperparameters described below, by running grid searches over a range of hyperparameter values and selecting the hyperparameter configuration with lowest error on the validation set. Correspondingly, the relative L^2 test errors of these models, with the best performing hyperparameters are reported in tables 6, 7 and 8.

Table 5: The left columns describe the grid resolution on which training (for a the regular grid) is performed. The right columns show the grid resolution on which inference or testing (for all coordinate configurations) is performed. Note, for the Navier-Stokes problem the input to the FNO must be available on a regular 65×65 grid, this is not the case for the DeepONet and VIDON.

Problem	*Training Grid*		*Test Grid*	
	space	time	space	time
Darcy Flow	51×51	-	51×51	-
Allen-Cahn	26×26	21	76×76	41
Navier-Stokes	33×33	-	65×65	-

We normalize the input and output data of the Darcy Flow and Navier-Stokes' problems to lie in the range $[0, 1]$. The data of the Allen-Cahn problem is already in this range and thus needs no additional normalization.

Each training was run on one of the following GPUs: Nvidia GTX 1080, Nvidia GTX 1080 Ti, Nvidia Tesla V100, Nvidia RTX 2080 Ti, Nvidia Titan RTX, Nvidia Quadro RTX 6000, Nvidia Tesla A100.

In the following we describe the detailed results (including error bars) and the training parameters that were used to obtain these results. Exact information on the training settings can also be obtained from the settings files in the accompanying code.

C.2 Detailed Results

Table 6: The mean $\pm$ standard deviation of relative test errors in $L^2(D)$ for the Darcy flow problem for different configurations of sensors. The symbol "-" implies that the model could not be used for this configuration.

Configuration	# (Sensors)	**FNO**	**DeepONet**	**VIDON**
Regular Grid	51×51	$0.76\% \pm 0.02\%$	$1.48\% \pm 0.01\%$	$1.29\% \pm 0.02\%$
Irregular Grid	$51^2 = 2601$	-	$1.52\% \pm 0.02\%$	$1.48\% \pm 0.07\%$
Missing Data	$[2081, 2601]$	-	-	$1.77\% \pm 0.01\%$
Perturbed Grid	[2341,2861]	-	-	$1.68\% \pm 0.01\%$
Random Locations	2601	-	-	$2.58\% \pm 0.01\%$
Variable Random Locations	[2341,2861]	-	-	$2.55\% \pm 0.01\%$

Table 7: Mean $\pm$ standard deviation of the relative test errors in $L^2(D \times (0, T))$ for the Allen-Cahn PDE for different configurations of sensors. The symbol "-" implies that the model could not be used for this configuration.

Configuration	# (Sensors)	**DeepONet**	**VIDON**
Regular Grid	26×26	$0.34\% \pm 0.01\%$	$0.26\% \pm 0.02\%$
Irregular Grid	$26^2 = 676$	$0.34\% \pm 0.01\%$	$0.27\% \pm 0.01\%$
Missing Data	$[541, 676]$	-	$0.63\% \pm 0.02\%$
Perturbed Grid	[608,744]	-	$0.83\% \pm 0.02\%$
Random Locations	676	-	$1.21\% \pm 0.03\%$
Variable Random Locations	[608,744]	-	$1.20\% \pm 0.03\%$

C.3 FNO Training Parameters

We use the implementation of the FNO model provided by the authors of [16] with some slight adjustments to make the code compatible with ours.

Table 8: Mean $\pm$ standard deviation of relative test errors in $L^2(D)$ for the Navier-Stokes PDE for different configurations of sensors. The symbol "-" implies that the model could not be used for this configuration.

Configuration	# (Sensors)	**FNO**	**DeepONet**	**VIDON**
Regular Grid	33×33	$3.49\% \pm 0.09\%$	$4.20\% \pm 0.02\%$	$5.22\% \pm 0.12\%$
Irregular Grid	$33^2 = 1089$	-	$4.33\% \pm 0.04\%$	$5.45\% \pm 0.09\%$
Missing Data	$[871, 1089]$	-	-	$5.64\% \pm 0.03\%$
Perturbed Grid	[980, 1198]	-	-	$5.34\% \pm 0.02\%$
Random Locations	1089	-	-	$8.35\% \pm 0.03\%$
Variable Random Locations	[980, 1198]	-	-	$8.28\% \pm 0.03\%$

The FNO model for both Darcy Flow and the Navier-Stokes equations was trained using 12 modes and width 32. The initial learning rate was set to 1e-3 and is halved every 100 epochs. Weight decay is set to 1e-8 and training finishes after 500 epochs. It is well-known that the training time per epoch for FNO can be significantly higher than that of DeepONet, however, FNO trains much faster i.e., with significantly fewer epochs [16, 22].

C.4 DeepONet Training Parameters

Table 9 shows the model sizes used for each problem and for each of the two applicable coordinate configurations (Regular and irregular grids). Note, we use our own implementation of the DeepONet. We run all trainings for a maximum of 100,000 epochs. Table 10 shows additional training parameters.

Table 9: Model sizes of the DeepONet used on each of the problems and for each of the two coordinate configurations (regular and irregular grids). The notation [20, 20, 20] refers to three layers with 20 neurons each.

Problem	p	Branch Net	Trunk Net
Darcy Flow	100	[250, 250, 250, 250]	[250, 250, 250, 250]
Allen-Cahn	400	[400, 400, 400, 400]	[500, 500, 500, 500]
Navier-Stokes	100	[250, 250, 250, 250]	[250, 250, 250, 250]

Table 10: Training parameters of the DeepONets used on each of the the problems and for each of the two coordinate configurations (regular and irregular grids). The third column indicates the epochs at which the learning rate is halved.

Problem	Initial Learning Rate	Halved at Epochs	Weight Decay
Darcy Flow	1e-4	5k, 30k, 60k, 90k	1e-9
Allen-Cahn	1e-4	20k, 40k, 60k, 80k	1e-9
Navier-Stokes	2e-4	5k, 30k, 60k, 90k	1e-7

C.5 VIDON Training Parameters

The following parameters are used in all trainings for all problems and coordinate configurations. The number of heads H in VIDON (2.6) was set to 4 for each configuration. The coordinate and sensor encodings in all trainings is set to four layers with 40 neurons each. The MLPs computing weights and values in VIDON (2.6) have four layers of 128 neurons each. The MLP which combines the concatenated output of each of the heads has four layers with 256 neurons each. The remaining model parameters are presented in Table 11 and additional training parameters are shown in Table 12. Moreover, the runtime between different sensor configurations, reported in Table 8 varies significantly because in the first three, the trunk net has to be evaluated at significantly fewer points in the output

domain than in the remaining configurations (where a large number of different, randomly chosen points has to be evaluated).

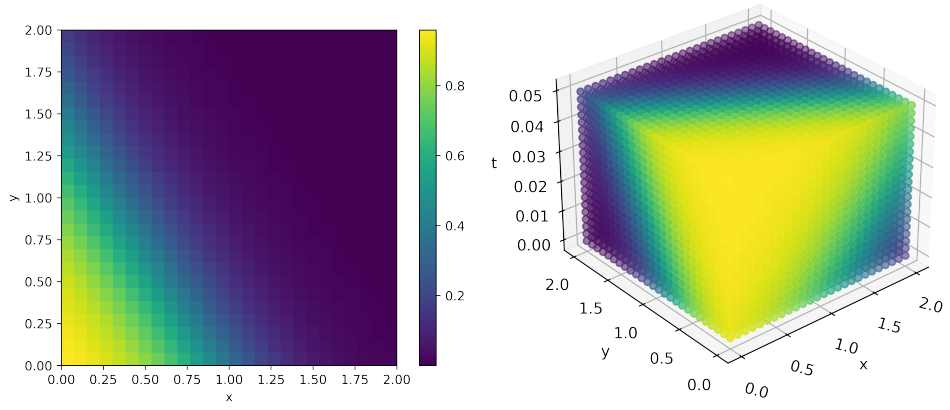
Table 11: Model sizes used on each of the problems and for both of the two coordinate configurations (regular and irregular grids). The notation [20, 20, 20] refers to three layers with 20 neurons each.

Problem	p	Output Neurons per Head	Trunk Net
Darcy Flow	100	64	[250, 250, 250, 250]
Allen-Cahn	400	64	[500, 500, 500, 500]
Navier-Stokes	100	32	[250, 250, 250, 250]

Table 12: Training parameters of VIDON used on each of the problems and for both of the two coordinate configurations (regular and irregular grids).

Problem	Initial Learning Rate	Halved at Epochs	Weight Decay
Darcy Flow			
Default	1e-4	20k, 40k, 60k, 80k	1e-9
Random / Variable Random Locations	1e-4	20k, 40k, 60k, 80k	1e-8
Allen-Cahn			
Default	1e-4	20k, 40k, 60k, 80k	1e-9
Missing, Variable Random Locations	2e-4	20k, 40k, 60k, 80k	1e-9
Navier-Stokes			
Default	1e-4	10k, 20k, 40k, 60k, 80k	1e-7
Random Locations	2e-4	10k, 20k, 40k, 60k, 80k	1.5e-7
Variable Random Locations	2e-4	10k, 20k, 40k, 60k, 80k	2e-7

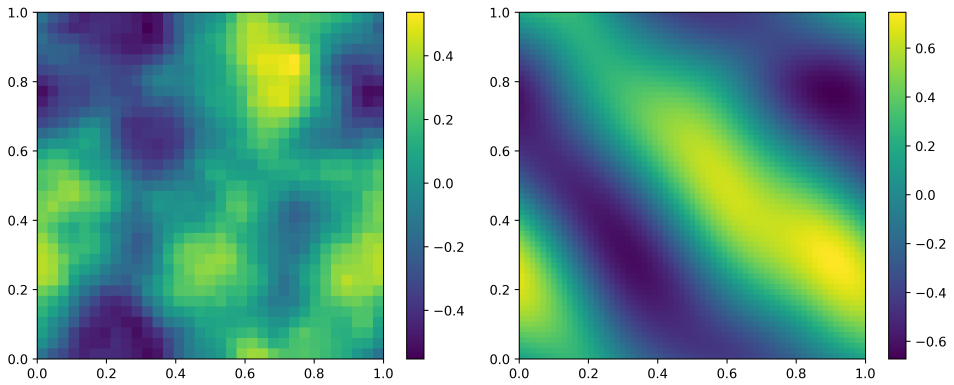
C.6 Additional Figures



(a) Input

(b) Output

Figure 4: A sample illustrating the solution operator for the Allen-Cahn equation (3.5). The input is given by the initial conditions and the output is given by the time-history (up to time $T = 0.05$) of the rotated travelling-wave solution (4.1) of the PDE.



(a) Input (b) Output

Figure 5: A sample illustrating the operator for the Navier-Stokes equations (3.7). The input to the operator is given by the initial vorticity and the output is the vorticity at time $T = 5$.