nature reviews physics https://doi.org/10.1038/s42254-024-00712-5

Perspective

Neural operators for accelerating scientific simulations and design

Kamyar Azizzadenesheli[1], Nikola Kovachki[1], Zongyi Li[2], Miguel Liu-Schiaffini[2], Jean Kossaifi[1] & Anima Anandkumar[2]

Abstract

Scientific discovery and engineering design are currently limited by the time and cost of physical experiments. Numerical simulations are an alternative approach but are usually intractable for complex real-world problems. Artificial intelligence promises a solution through fast data-driven surrogate models. In particular, neural operators present a principled framework for learning mappings between functions defined on continuous domains, such as spatiotemporal processes and partial differential equations. Neural operators can extrapolate and predict solutions at new locations unseen during training. They can be integrated with physics and other domain constraints enforced at finer resolutions to obtain high-fidelity solutions and good generalization. Neural operators are differentiable, so they can directly optimize parameters for inverse design and other inverse problems. Neural operators can therefore augment, or even replace, existing numerical simulators in many applications, such as computational fluid dynamics, weather forecasting and material modelling, providing speedups of four to five orders of magnitude.

Sections

[1]NVIDIA, Santa Clara, CA, USA. [2]Caltech, Pasadena, CA, USA. e-mail: anima@caltech.edu

Introduction

The scientific method consists of a set of hypotheses formed through inductive reasoning based on observations, which are first systematically tested before being iteratively updated. However, testing hypotheses in many scientific domains involves slow and expensive real-world experiments, which limits scientists to a small set of hypotheses that are driven by prior knowledge and intuition. If expensive experiments could be replaced with faithful computational simulations, that would vastly expand the hypothesis space for further exploration.

In principle, it is possible to create such simulations when the (approximate) governing laws of many physical, chemical and engineering processes are known. The physical world around us is often modelled by governing partial differential equations (PDEs) derived from first principles, such as the laws of thermodynamics, chemical interactions and mechanics[1]. For example, the aerodynamics of cars and planes can be modelled using the Navier–Stokes PDE defined on the geometry of the surface[2]. However, it is currently infeasible to run numerical simulations at the scale and fidelity needed to entirely replace physical experimentation. This is because existing numerical methods used in simulations require a fine computational grid to guarantee convergence of the iterations and the cost of computing increases at least quadratically with the resolution of the grid.

In climate modelling, for example, PDEs are used to simulate the future of Earth's climate, which is critical for policy-making and developing strategies for climate-change mitigation. In this case, physical experimentation is impossible, so simulations are the only option. However, current numerical methods require massive computing power to resolve the physics at finer scales where the effects of storms and cloud turbulence can be faithfully reproduced, and even the world's most powerful supercomputers do not suffice[3]. Moreover, numerical methods have other shortcomings pertinent to applications involving observational data where the governing laws are only partially specified or inaccurate. Current numerical methods cannot directly exploit observational data for such cases, as they are not explicitly data-driven. Additionally, numerical methods are not generally differentiable, thus making inverse problems and design optimization expensive and slow[4].

Artificial intelligence (AI) is now transforming the ability to create fast, high-fidelity simulations of many complex multiscale processes[5–7]. Previous developments such as sparse representation[8], recurrent neural networks[9], reservoir computing[10] and others have already shown promise in modelling dynamical systems. The latest advances in AI, termed neural operators[11,12], can model such phenomena and capture finer scales, leading to high-fidelity solutions. Neural operators are a generalization of neural networks that can learn operators: mappings between two functions defined on continuous domains. They have several advantages. First, they can serve as fast and accurate surrogates for numerical methods and overcome the high computational requirements of numerical methods. Unlike conventional numerical methods, neural operators can learn an efficient feature representation, or basis, that does not require a fine grid. Further, once the neural operator model is trained, the inference is typically a single forward pass, as opposed to a large number of iterations required in numerical methods. In addition, because neural operators are AI models, they can build on the latest hardware and software advances of accelerated computing[13] to obtain the best possible speedups with the learned representations. This has resulted in 45,000× speedup in the weather forecast[14], 26,000× in the automotive industry[15] and 700,000× in the prediction of carbon dioxide geological storage[16], along with many other scientific computing domains. Second, neural operators can learn directly from observational data and do not suffer from modelling errors present in numerical methods that, for many applications, rely solely on simplified equations. Third, neural operators are deep learning models implemented using deep learning packages, so they are differentiable by design and can be used directly for inverse problems such as optimized design generation. In contrast, traditional solvers are usually not differentiable and require many iterations using methods such as Markov chain Monte Carlo. Fourth, neural operators are democratizing science as running the trained neural operators does not require the deep domain expertise needed to set up and run many traditional solvers. Moreover, many applications, such as weather forecasting, that previously required large central processing unit clusters can now be carried out using consumer desktop graphics processing units and open-source models, enabling widespread accessibility[14,17,18].

In this Perspective, we introduce neural operators, discuss their formulation and applications, and argue that they will have a transformative impact in many applications.

Neural networks and neural operators

AI models such as transformers and convolutional neural networks have shown impressive performance in domains such as text and images[19]. However, in our opinion, standard neural networks have fundamental limitations that make them unsuitable for scientific modelling. First, they have limited generalization beyond conditions seen during training. In particular, they are limited to learning and making predictions at the resolution of the training data. This is a considerable limitation because many scientific phenomena occur on continuous domains (for example fluid flows, wave propagation and material deformation) even though only discretized observations are available for training. A simple solution to the above issue that enables pretrained neural networks to predict at any resolution is to use standard interpolation methods, such as bilinear interpolation on the predictions made at locations on the training grid. However, such approaches perform poorly on many processes (for example fluid flows) as they fail to capture finer scales accurately (see the discussion of Fig. 1 below)[20]. This is because simple interpolation schemes cannot capture scales that are not present in limited-resolution training data.

Second, in many applications, the availability of high-resolution data may be limited, and combining it with larger amounts of low-resolution data during training is not feasible in standard neural networks, which only support fixed-resolution inputs and outputs. Even if modifications are made to incorporate multiple resolutions, such as for climate modelling datasets[21], at inputs or outputs[22], these modifications are not principled and have no guarantees for capturing finer scales.

More generally, neural networks only learn mappings between inputs and outputs of fixed dimensions (Fig. 2a); they do not have the expressive power to capture mappings between functions on continuous domains, known as operators. In contrast, neural operators can predict the solution at any location in the output domain and are not limited to the grid of training data[12,23] (Fig. 2b). They accomplish this by approximating the underlying operator, which is the mapping between the input and output function spaces, each of which can be infinite-dimensional (Fig. 2c). In other words, neural operators approximate the continuum mapping even when trained only on discrete data, and thus they can capture the finer scales more faithfully.

To illustrate the above, Fig. 1 shows the prediction of a neural operator, the Fourier neural operator (FNO), for fluid dynamics[11], and of UNet, a popular neural network trained at a fixed resolution and then augmented with trilinear interpolation. Figure 1 depicts the long-term

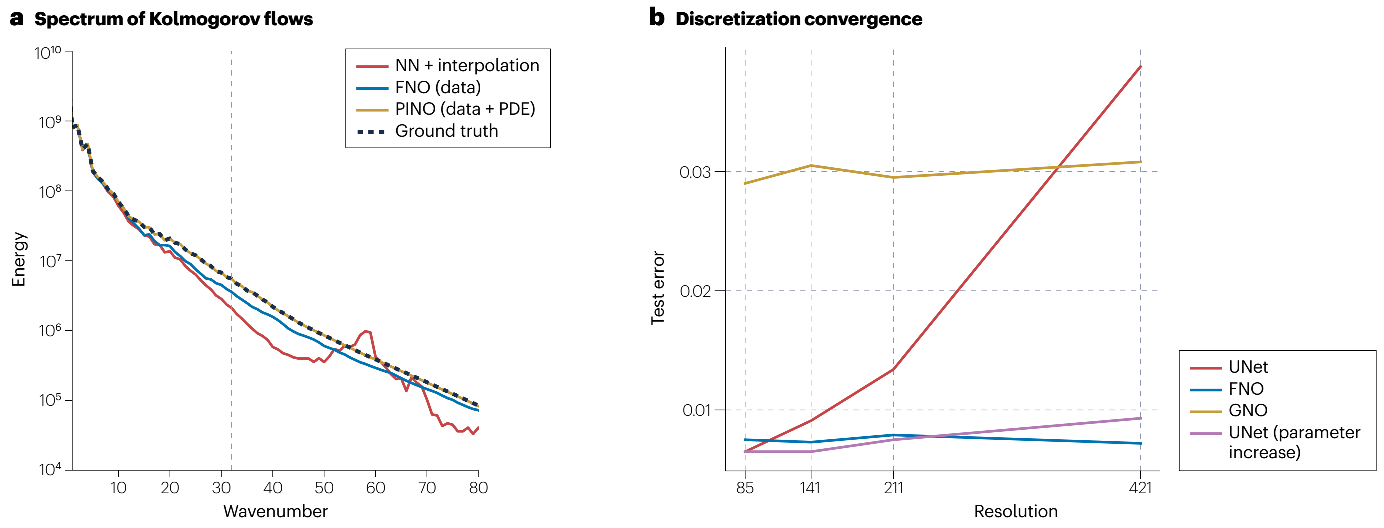

Fig. 1 | Spectral overall analysis. a, The *x*-axis is the Fourier wavenumber and *y*-axis is the energy per spectrum. In the isotropic Navier–Stokes equation (Kolmogorov flows), the cornerstone of microscale turbulent flow, Fourier neural operators (FNOs) can extrapolate to unseen frequencies using only limited-resolution training data[98]. The low frequencies to the left of the vertical dashed line are the training frequency resolution, and the models extrapolate to higher frequencies on the right of the dashed line. A physics-informed neural operator (PINO) uses both training data and the partial differential equation (PDE) for the loss function, and can perfectly recover the ground-truth spectrum. A trained UNet (a popular neural network) with trilinear interpolation (NN + interpolation) has severe distortions at higher frequencies, beyond the resolution of training data. **b**, Resolution invariance. The *x*-axis is the resolution of the test data, and the *y*-axis is the test error at that given resolution. Neural operators are discretization-convergent, meaning that the model converges to the target continuum operator as the discretization is refined. This is illustrated using the Darcy equation which describes a fluid flow in a porous medium. Each architecture – UNet, FNO and graph neural operator (GNO) – was trained at a given resolution and tested at that same resolution[12] (no superresolution). FNO and GNO have consistent errors as resolution increases, but UNet has increasing errors because the size of its receptive field changes with resolution, and it does not enjoy the guarantees of discretization convergence. UNet is only able to maintain the same test error as the resolution increases if the number of parameters increases. In this case, the convolutional filter size was increased for resolutions 85, 141, 211 and 421, corresponding to about 2.2M, 6.0M, 11.8M and 19.3M parameters, respectively.

statistics of interest, that is, the spectrum of the fluid, representing the per frequency characteristics of the chaotic system, emulated using the trained models for the benchmark Kolmogorov chaotic flow. FNO follows the trend of ground-truth energy decay in the frequency domain, even beyond the training frequencies, whereas UNet with interpolation fails to predict the finer scales. The behaviour seen in Fig. 1b demonstrates that empirically, FNOs and other neural operators can do zero-shot superresolution and extrapolate to unseen higher frequencies, demonstrated on the benchmark stationary Darcy flow equation. In contrast, as discussed elsewhere[24], band-limited operators and representation-equivalent operators[25], such as spectral neural operators[26], cannot generate new (higher) frequencies because their representation space is fixed. Therefore, they introduce an irreducible approximation error based on the size of the predefined representation space. As the goal of operator learning is to find the underlying solution operator in the continuum, we believe that discretization convergence is a more useful strategy. However, the error cannot be expected to be (nearly) zero, as the model, during training, cannot access information at the finer scales owing to the limited resolution of the samples. In scientific modelling, one usually has access to physics constraints such as PDEs that fully constrain the system or conservation laws and symmetries that partially specify the system. The physics-informed neural operator (PINO)[24] incorporates both data and physics losses, leads to even better extrapolation to higher frequencies and almost perfectly matches the ground-truth spectrum in Fig. 1a.

Formulation

Neural operators learn operators and thus can predict at any point in the continuous domain. In the supervised learning setting, the input and output functions are discretized to obtain the training data. For instance, in the case of weather forecasting, the input consists of variables such as temperature and wind speed on the Earth's surface and in the atmosphere, whereas the output represents the values of the same variables at a later time. Although the variables are defined on the continuous space-time domain, the training data are available only at a certain discretization – for example, in the ERA-5 reanalysis dataset, it is 25 km and hourly intervals[27]. The goal of neural operators is to make predictions on continuous domains, despite the available training data being discrete.

Design of a neural operator

Standard neural networks consist of a series of blocks that perform linear functions, such as fully connected or convolutional layers, followed by nonlinear activations such as rectified linear units. Neural operators[12] follow the same philosophy, that is, linear blocks followed by nonlinear transformations, the difference being that linear blocks in neural operators are realized by integral operators instead of linear functions in fixed dimensions. Similar to the neural network class that subsumes general linear models as their single-layer special case, the neural operator class subsumes linear integral operators, such as solution operators to linear PDEs, commonly realized using Green's

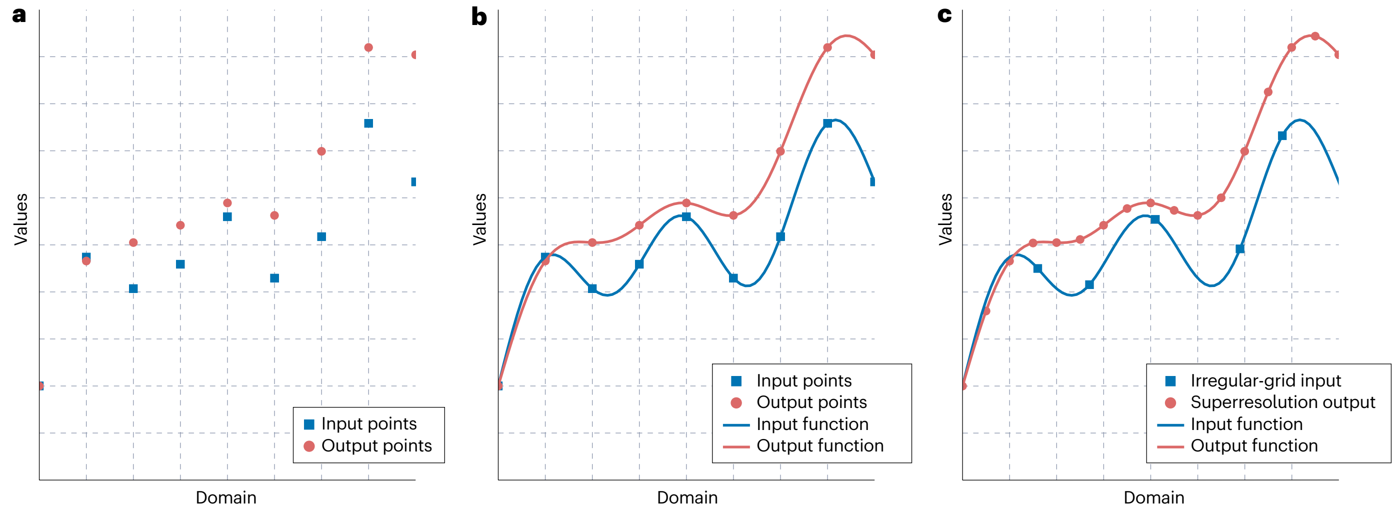

Fig. 2 | Comparison of neural networks with neural operators. a, A neural network learns a mapping between input and output points on a fixed, discrete grid. **b**, A neural operator maps between functions on continuous domains, even when training data are on a fixed grid. **c**, Because a neural operator maps between functions, it accepts inputs outside the training grid and can do superresolution. Figure courtesy of Benedikt Jenik.

function integration, as their single-layered special case. The linear integral operator is given by

$$\int \kappa(x,y)v(y)\mathrm{d}y \approx \sum_{i}^{N} \kappa(x,y_i)v(y_i)\Delta y_i, \qquad (1)$$

where $v(\cdot)$ is the input function to the operator block, and $\kappa(x,y)$ denotes a learnable kernel between any two points x and y in the output and input domains, respectively. A neural operator consists of linear integral operator blocks given by equation (1), followed by pointwise nonlinear activations, for example Gaussian error linear units (GeLU)[23]. Rectified linear units (ReLU) are usually avoided because of non-smoothness[28]. The expressiveness and capacity of the model are determined by the number of layers and the expressivity of the mentioned submodels, for which domain experts' knowledge is advised for proper designs, proper fit and avoiding overfitting. See Fig. 3 for the detailed architecture of layers of integral operators.

Note that the query point x in the output domain in equation (1) need not be limited to the discrete grid of the training data and can be any point in the continuous domain. Similarly, the input discretization, that is, grid points y_i, can be chosen arbitrarily on which the input function $a(\cdot)$ is specified, and as the discretization becomes finer (N in the Riemannian sum (1) grows larger), this approximation becomes more accurate. Neural operators can thus take input functions at any discretization and be queried for predicting solutions at any point in the domain, even if the query point is outside the training grid. Neural operators possess a property known as discretization convergence, meaning that they converge to a unique operator in the limit of mesh refinement[12]. This is a crucial characteristic that is required of any sound approach to scientific computing and is possessed by daily used integral approximation methods such as Riemannian sum. Intuitively, this characteristic follows from the integral approximation method, such as Riemannian sum, Galerkin or Fourier spectral, used in neural operators. Neural networks do not have such guarantees, even when augmented with interpolation methods such as bilinear interpolation.

Zero-shot superresolution and evaluation
Two properties follow directly from the discretization convergence property: first, zero-shot superresolution, where a trained neural operator can receive an input sample at a given resolution and output a prediction at a higher resolution than what was provided during training; second, zero-shot superevaluation, where a trained neural operator can be evaluated on a new, finer discretization than what was used during training and produce a solution on that new resolution, thus potentially resolving higher-frequency details.

Parameterization of the kernel
Although there are several choices to parameterize equation (1), as outlined below, the choice of nonlinear activation is standard – for example, Gaussian error linear units. In addition, neural operators include pointwise lifting and projection operations. The lifting operator encodes the input function values to a very high dimension, greatly increasing the model expressivity and allowing integration operation in a very high dimension. The projection operator decodes the high-dimensional representation to the function dimension and projects the feature representation to the native output dimension. This is especially important when the input and output functions are scalar-valued or have a low-dimensional range space. If the kernel is learned in such low dimensions, it has limited expressivity, and the lifting operation expands it into arbitrary channels, enriching the model capacity and expressivity[29]. The lifting and projection operations are pointwise and can be linear or nonlinear; however, nonlinear operations are, in general, more expressive[30]. Given an architecture and training data, neural operators can be trained end-to-end using standard algorithms such as SGD or Adam[31], similar to neural networks.

Architectures
Many architectures have been proposed to parameterize the linear integral operation in equation (1), some drawing inspiration directly from existing numerical methods: for example, the kernel $\kappa(x,y)$ in equation (1) is constrained to be separable in x and y, yielding the DeepONet

model[32]. Although the original DeepONet model is constrained to be on a fixed input grid, later extensions of DeepONet[33,34] remove this limitation, which preserves discretization convergence, and thus is a special case of neural operators.

Graph neural operators (GNO)[23] are architectures that are operator extensions of graph neural networks[35] and support the kernel integration on a fixed-radius ball. When the ground-truth operator is non-local, GNO has limited expressivity when the radius is small, as it cannot capture the global effects, owing to a limited receptive field. Using GNO becomes computationally expensive if the receptive field is increased and the radius for the neighbourhood becomes large. To overcome this problem and to fit a non-local operator, hierarchical or multilevel extensions have been proposed[36], inspired by the multipole decomposition of the kernel and multigrid iterative solvers[37,38].

FNO[11] is an alternative approach to obtaining a global operator. In FNO, the kernel integration in equation (1) is solved through element-wise multiplication in the Fourier domain with learned coefficients and inverse Fourier transform back to the original domain. The FNO approach is inspired by the pseudospectral solvers[39] (Fig. 3) for which the Fourier basis is used, and operations are iteratively carried, as shown in Fig. 3. Although inspired by this approach, FNO has nonlinear components to enhance its expressivity. FNO emulates the pseudospectral solvers efficiently[40]. Moreover, Fourier domain learning is a good inductive bias in many domains, such as fluid dynamics. For example, it is well known that in the Navier–Stokes equation the energy cascades from low-frequency modes to the high frequencies owing to the advection, which is subsequently killed off by dissipation[41]. Similar, yet more complicated, dynamics are exhibited in the Navier–Stokes and Euler's equations[42]. By learning directly in Fourier space, the FNO allows us to capture such dynamics efficiently. Extensive theoretical analyses have been developed for neural operators[29,40]. It has been shown that neural operators, in particular FNOs, are universal approximators of continuous operators[12,40], generalizing the classical results[43] to modern deep learning.

FNO is computationally efficient for capturing global dependencies when the input is limited to a regular grid, as the fast Fourier transform can be computed with quasilinear complexity. For irregular grids, however, computing the Fourier transform has quadratic complexity, which scales poorly for large problems. Such a Fourier transform on an irregular grid is proposed in the Vandermonde neural operator[44] that carries the non-uniform discrete Fourier transform approximation, uses the low mode property of FNO and promotes the matrix form of Fourier transform approximation as opposed to fast Fourier transform to gain computation advances. To enhance computational efficiency, more recent methods instead propose deformation[45] or graph kernels[15] to transform the irregular physical grid to a regular latent grid on which fast Fourier transform is applied. Further developments include decomposing the kernel integration in the spatial domain, where the grid is often irregular, while keeping a regular grid in the temporal domain, for example for seismic monitoring[46] and for incorporating symmetries in physics[47]. Such methods apply GNOs on the spatial domain and FNOs on the temporal domain, resulting in efficient architectures[46]. Other developments include tensorized Fourier neural operators[48] and the U-shaped neural operator[49] to efficiently parameterize the Fourier domain.

Other works have proposed convolutional neural operators for convolutions directly in the spatial domain[50] and thus extend convolutional neural networks to operator learning. It was further suggested[12] that the linear integral transform κ in equation (1) can be extended to nonlinear integral transforms, such as $\kappa(x,y,a)$, where the kernel can vary with the input function $a(\cdot)$. One such example is the transformer architecture[19]. For the special case of a fixed grid with shared input and output domain, the kernel has the form $\kappa(x,y,a) := \exp\langle W^Q a(x), W^K a(y)\rangle/\sqrt{m}$, where m is the inner dimension of the self-attention layer, and the integration is defined as

$$\int \mathcal{S}(\kappa(x,y,a))W^V a(y)\mathrm{d}y = \int \mathcal{S}\left(\frac{\exp\langle W^Q a(x), W^K a(y)\rangle}{\sqrt{m}}\right)W^V a(y)\mathrm{d}y \quad (2)$$

where $\mathcal{S}$ is the softmax function and W^Q, W^K and W^V are the query, key and value matrices of transformer architectures[51]. Whereas the standard transformer in equation (2) is limited to a fixed grid, various attempts have been made to turn it into an operator and allow prediction on irregular grids, for example the operator transformer (OFormer)[52], the mesh-independent neural operator (MINO)[53] and the general neural operator transformer (GNOT)[54]. However, these approaches become computationally intractable because the attention mechanism scales quadratically with the input resolution. To alleviate this problem, in vision transformers[55,56], patches are used to reduce dimensionality, but the choice of fixed-size patches limits the model to a fixed resolution and does not yield an operator. It remains an open problem how to develop scalable and expressive transformer architectures for operator learning.

Implicit neural representation

The idea of using neural networks to represent continuous functions, known as implicit neural networks, has a rich history[57]. One example is when the solution function of a single instance of PDE is represented through a neural network, known as the physics-informed neural network (PINN)[58–61]. Similarly, in computer vision and graphics, neural

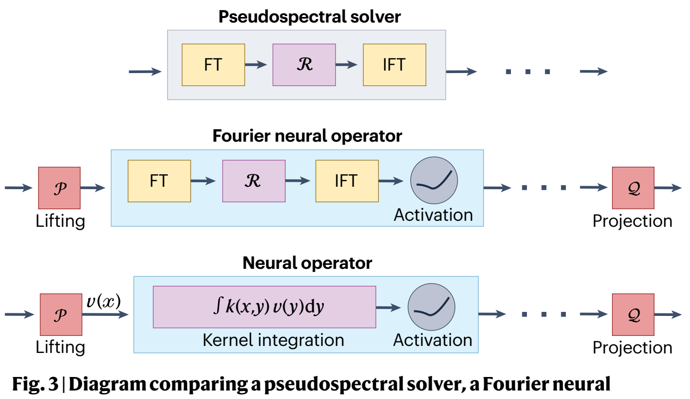

Fig. 3 | Diagram comparing a pseudospectral solver, a Fourier neural operator and the general neural operator architecture. In general, lifting and projection operators $\mathcal{P}$, $\mathcal{Q}$ can be nonlinear. Pseudospectral solvers are popular numerical solvers for fluid dynamics where the Fourier basis is used, and operations are iteratively carried out, as illustrated. In spectral methods, the integration is carried in the Fourier domain as a mean of multiplication following the Fourier theorem, for which the kernel is also represented in the Fourier domain using a design matrix R. The Fourier neural operator (FNO) is inspired by the pseudospectral solver, but has a nonlinear representation that is learned, and the design matrix R is also made trainable. FNO is a special case of the neural operator framework, shown on the last row, where the kernel integration is carried out through different methods, such as direct discretization or the Fourier transform. κ refers to the kernel that parameterizes the integral operator and v is the input function to the first integral layer (equation (1)). FT and IFT refer to Fourier and Inverse Fourier Transforms.

radiance fields[62–64] have been used to represent 3D scenes. Note that in both these cases, the implicit neural network is used to represent a single function, for example a single velocity field or a 3D scene. In contrast, neural operators can generate the output functions for arbitrary input functions. Thus, fundamentally, neural operators are a strict generalization of implicit neural networks: from representing a single function to learning operators. In other words, neural operators can be viewed as conditional neural fields, conditioned on different input functions. From this perspective, there have been several other attempts to extend implicit neural networks to handle multiple instances by additionally learning a parameterization over the space of input functions[65]. However, in general, this mapping is not well posed, because one function can be potentially mapped to multiple network parameterizations. To overcome this problem, recent works use a reduced-order model that maps each input function to a small number of latent parameters (codes) and then feeds it as input to a neural field, similar to an encoder–decoder model[66,67]. In contrast, a neural operator is a more direct way to handle multiple instances of input and output functions through operator learning.

Physics-informed neural operator

PINNs and variants[68] have been effective in solving steady-state PDEs[69–72]. However, the optimization landscape of PINNs becomes challenging for problems such as time-varying PDEs, and the solution cannot be easily found using standard gradient-based methods[73]. This issue can be alleviated by extending PINN to a physics-informed neural operator (PINO)[24,74,75], which incorporates training data in addition to PDE information, making the optimization landscape more tractable and enables the learning of the solution operator of complex time-varying PDEs.

Moreover, PINO has superior generalization and extrapolation capabilities, and reduced training data requirements, when compared with purely data-driven neural operators. It is possible to further improve the test-time accuracy of PINO, by fine-tuning the model to minimize the PDE loss function, on the given PDE instance at test time. This fine-tuning step is identical to the PINN optimization process but uses the initialization from a pretrained PINO model and fine-tunes it further, instead of optimization from a random initialization carried out in PINN. Thus, it can overcome the optimization difficulties in PINN.

In PINO[24], one can combine low-resolution training data with high-resolution physics constraints to learn a good approximation of the underlying operator and carry out accurate zero-shot super-resolution. Figure 1 shows results after training with the FNO backbone on low-resolution data (64×64) and further fine-tuned using PDE loss at higher resolution (PINO) and tested on high-resolution data (256×256). We believe that having such multiresolution losses is crucial to obtaining high-fidelity learned solvers that can resolve fine-scale features.

An alternative set of techniques is motivated by the Monte Carlo methods, where one can leverage the probabilistic representation of PDEs to obtain suitable objectives for deep learning. Specifically, reformulations of PDEs based on the Feynman–Kac formula and backward stochastic differential equations have been successfully used to tackle high-dimensional parabolic and elliptic PDEs[70,76–79]. Recently, such ideas have been combined with neural operators, promising efficient, derivative-free alternatives to PINOs[80].

Generative neural operators

We have so far discussed neural operators for approximating deterministic mappings between function spaces. There are also extensions to learning probabilistic mappings on function spaces. These are either based on generative adversarial networks[81], diffusion models[82], or variational autoencoders[83] that are extended to function spaces. In particular, the diffusion neural operator learns a score operator that takes a Gaussian random field as input and can generate samples at any resolution specified. This has been applied to scientific problems where many samples of possible real-world phenomena outcomes are of interest. Among these are volcanic and seismic activities[81,84], statistics of Navier–Stokes equations and resolution-free visual sensors[82].

Applications

Neural operators have yielded impressive speedups over traditional numerical solvers in a wide range of domains. They can help achieve medium-range weather forecasting tens of thousands of times faster than with current numerical weather models and were the first data-driven approach that resulted in accurate high-resolution (0.25 degrees) weather forecasting[14,85]. This speedup enables accurate risk assessment of extreme weather events such as hurricanes and heat waves that requires many forward runs of weather models for uncertainty qualification, for example ensemble methods. An improved neural operator version based on the spherical FNO[17] is now running on an experimental basis alongside traditional forecasting systems at the European Centre for Medium-Range Weather Forecasts[86]. Similarly, for the application of carbon capture and storage for climate-change mitigation, the nested FNO model is hundreds of thousands of times faster, enabling large-scale assessment of reservoirs for geological CO_2 storage[16]. The speedup enables a rigorous probabilistic assessment for maximum pressure buildup and carbon dioxide plume footprint, with a runtime of only 2.8 seconds which would otherwise have taken nearly 2 years with numerical simulators. Other applications include simulations of fluid dynamics[87–89], 3D industrial-scale automotive aerodynamics[15], 3D dynamic urban microclimate[90], material deformation[45,91,92], computational lithography[93,94], photoacoustics[95] and electromagnetic fields[96,97]. Further, neural operators have been used for learning long-term statistical properties such as attractors[98,99] in chaotic systems and detecting tipping points in non-stationary systems[100]. Neural operators based on generative adversarial networks[81] or diffusion models[82] in function spaces can be used in the modelling of stochastic natural phenomena (such as volcanic[101] and climate activities[102]) and stochastic differential equations[103]. Researchers have also explored building hybrid solvers by using neural operators combined with numerical solvers[104].

Neural operators are also effective for inverse problems, where the aim is to find a parameter from noisy observations, and in many cases, the forward model is non-invertible or has a poorly conditioned inverse. Examples include inverse design, optimization, control[105] and risk assessment.

Because neural operator models offer fast inference and can be efficiently differentiated, they can be used either for sampling the Bayesian solution to an inverse problem through Markov chain Monte Carlo methods[106] or for obtaining a point estimate by optimizing a regularized functional[107]. An invertible neural operator framework[108] has been introduced for such problems. In inverse design, the goal is to optimize over a given set of parameters, given a forward model, and provide guidance for iterative design improvements or generate optimized designs from scratch. For example, by using a neural operator model it was possible to produce an optimized design for a new medical catheter that reduces bacterial contamination by two orders of magnitude[109]. The neural operator model could accurately simulate the bacterial density in fluid flow and help to optimize design shapes within the catheter to prevent bacteria from swimming upstream into the human body. Neural operators are also effective for other inverse problems, for example in seismology, where, given the seismic waves and earthquakes observed in the subsurface of the Earth, one needs to infer the structure of the subsurface causing the specific seismic observation[110–113].

Software implementation and benchmarks

Efficient software and hardware implementation are crucial to successfully applying neural operators to practical problems. A reference implementation for various neural operators in PyTorch is available (see Code availability). In addition to libraries, standardized benchmarks are crucial for further development. However, benchmarking has been a challenge due to the disparate nature of different systems across scientific fields. For instance, the choice of PDE parameters (forcing functions, domain size, resolution of training data) can affect the emulated dynamics of the functions. There are two main categories of PDE benchmarks: first, broad, unified benchmarks for machine learning in physical systems[114–117] and second, task-specific or domain-specific benchmarks[118–123].

To aid the further development of neural operators, it is critical to construct more universal benchmarks that can correctly assess operator-learning capabilities. In our opinion, many current benchmarks do not meet these requirements, and many are limited to training and evaluation to a single resolution[116] or do not provide well-resolved training data for operator learning to be effective in the purely data-driven setting[116]. In addition to learning metrics, the benchmarks also need to incorporate hardware performance metrics that are typically used in high-performance computing studies[124] and explore the benefits of low-precision learning[125].

Outlook

We have shown that neural operators are a principled AI approach for modelling multiscale processes in various scientific domains, such as fluid dynamics, wave propagation and material properties. Multiscale processes, where microscopic interactions affect the macroscopic behaviour, span many scales of interactions. Current numerical methods are too expensive to simulate such processes faithfully, as they require iterating on a fine grid. Neural operators achieve notable speedups over current methods by learning nonlinear feature transformations and overcoming the need for a fine grid. In addition, they are naturally suited for inverse problems as they are differentiable, and one can directly invert the trained forward model through gradient-based optimization. In contrast, traditional simulations are non-differentiable and require expensive Markov chain Monte Carlo methods for inverse problems. Neural operators can be queried at any discretization and not just limited to the training resolution. However, as a modern deep learning model, neural operators are susceptible to overfitting if parameterized ineffectively. In minimal data settings, they may overfit the training data, and in very low-resolution scenarios, they may overfit the lack of resolution. In such cases, they may lack generalization to very different out-of-distribution data settings. Although these issues can be addressed through domain knowledge integration design, more data, higher fidelity data and physics equations, one should be mindful of them.

We believe that neural operators present a transformative approach to simulation and design, enabling rapid research and development.

Code availability

A reference implementation for various neural operators including and examples on how to get started can be found at: Neural Operator Library, https://github.com/neuraloperator/.

Published online: 08 April 2024

References

1. Evans, L. C. *Partial Differential Equations* Vol. 19 (American Mathematical Society, 2022).
2. Batchelor, G. K. *An Introduction to Fluid Dynamics* (Cambridge Univ. Press, 1967).
3. Schneider, T. et al. Climate goals and computing the future of clouds. *Nat. Clim. Change* **7**, 3–5 (2017).
4. Tarantola, A. *Inverse Problem Theory and Methods for Model Parameter Estimation* (Society for Industrial and Applied Mathematics, 2004).
5. Brunton, S. L., Noack, B. R. & Koumoutsakos, P. Machine learning for fluid mechanics. *Annu. Rev. Fluid Mech.* **52**, 477–508 (2020).
6. Kochkov, D. et al. Machine learning–accelerated computational fluid dynamics. *Proc. Natl Acad. Sci. USA* **118**, e2101784118 (2021).
7. Vinuesa, R. & Brunton, S. L. Enhancing computational fluid dynamics with machine learning. *Nat. Comput. Sci.* **2**, 358–366 (2022).
8. Brunton, S. L., Proctor, J. L. & Kutz, J. N. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proc. Natl Acad. Sci. USA* **113**, 3932–3937 (2016).
9. Vlachas, P. R., Byeon, W., Wan, Z. Y., Sapsis, T. P. & Koumoutsakos, P. Data-driven forecasting of high-dimensional chaotic systems with long short-term memory networks. *Proc. R. Soc. A* **474**, 20170844 (2018).
10. Pathak, J. et al. Hybrid forecasting of chaotic processes: using machine learning in conjunction with a knowledge-based model. *Chaos* **28**, 041101 (2018).
11. Li, Z. et al. Fourier neural operator for parametric partial differential equations. In *Proc. 9th International Conference on Learning Representations* (ICLR, 2021).
12. Kovachki, N. B. et al. Neural operator: learning maps between function spaces with applications to PDEs. *J. Mach. Learn. Res.* **24**, 1–97 (2023).
13. Dally, W. J., Keckler, S. W. & Kirk, D. B. Evolution of the graphics processing unit (GPU). *IEEE Micro* **41**, 42–51 (2021).
14. Pathak, J. et al. FourCastNet: a global data-driven high-resolution weather model using adaptive Fourier neural operators. In *Proc. Platform for Advanced Scientific Computing Conference (PASC)* (ACM, 2023).
15. Li, Z. et al. Geometry-informed neural operator for large-scale 3D PDEs. In *Advances in Neural Information Processing Systems 36* (NeurIPS, 2023).
16. Wen, G. et al. Real-time high-resolution CO_2 geological storage prediction using nested Fourier neural operators. *Energy Environ. Sci.* **16**, 1732–1741 (2023).
17. Bonev, B. et al. Spherical Fourier neural operators: learning stable dynamics on the sphere. In *Proc. 40th International Conference on Machine Learning* Vol. 202, 2806–2823 (PMLR, 2023).
18. Pathak J. et al. *Open-Source FourCastNet v2 Weather Model Hosted on ECMWF* (Github, 2023); https://github.com/ecmwf-lab/ai-models-fourcastnetv2.
19. Vaswani, A. et al. Attention is all you need. In *Advances in Neural Information Processing Systems* Vol. 30 6000–6010 (eds Guyon, I. et al.) (Curran Associates, 2017).
20. Esmaeilzadeh, S. et al. MeshfreeFlowNet: a physics-constrained deep continuous space-time super-resolution framework. In *SC'20: Proc. International Conference for High Performance Computing, Networking, Storage and Analysis* 1–15 (IEEE, 2020).
21. Haarsma, R. J. et al. High Resolution Model Intercomparison Project (HighResMIP v1. 0) for CMIP6. *Geosci. Model Dev.* **9**, 4185–4208 (2016).
22. Yuan, Y. et al. HRFormer: high-resolution vision transformer for dense predict. *Adv. Neural Inf. Process. Syst.* **34**, 7281–7293 (2021).
23. Li, Z. et al. Neural operator: graph kernel network for partial differential equations. *J. Mach. Learn. Res.* **24**, 1–97 (2023).
24. Li, Z. et al. Physics-informed neural operator for learning partial differential equations. *ACM J. Data Sci.* https://doi.org/10.1145/3648506 (2024).
25. Bartolucci, F. et al. Representation equivalent neural operators: a framework for alias-free operator learning. In *Advances in Neural Information Processing Systems 36* (NeurIPS 2023).
26. Fanaskov, V. & Oseledets, I. Spectral neural operators. *Dokl. Math.* https://doi.org/10.1134/S1064562423701107 (2024).
27. Hersbach, H. et al. The ERA5 global reanalysis. *Q. J. R. Meteorol. Soc.* **146**, 1999–2049 (2020).
28. Nair, V. & Hinton, G. E. Rectified linear units improve restricted Boltzmann machines. In *Proc. 27th International Conference on Machine Learning (ICML-10)* 807–814 (ICML, 2010).
29. Lanthaler, S., Li, Z. & Stuart, A. M. The nonlocal neural operator: universal approximation. Preprint at https://doi.org/10.48550/arXiv.2304.13221 (2023).
30. Lanthaler, S., Molinaro, R., Hadorn, P. & Mishra, S. Nonlinear reconstruction for operator learning of PDEs with discontinuities. In *11th International Conference on Learning Representations* https://openreview.net/forum?id=CrfhZAsJDsZ (ICLR, 2023).
31. Kingma, D. P. & Ba, J. Adam: A method for stochastic optimization. In *Proc. 3rd International Conference on Learning Representations* (ICLR 2015).

32. Lu, L., Jin, P., Pang, G., Zhang, Z. & Karniadakis, G. E. Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators. *Nat. Mach. Intell.* **3**, 218–229 (2021).
33. Lanthaler, S., Mishra, S. & Karniadakis, G. E. Error estimates for DeepONets: a deep learning framework in infinite dimensions. *Trans. Math. Appl.* **6**, tnac001 (2022).
34. Lu, L. et al. A comprehensive and fair comparison of two neural operators (with practical extensions) based on fair data. *Comput. Methods Appl. Mech. Eng.* **393**, 114778 (2022).
35. Battaglia, P. et al. Interaction networks for learning about objects, relations and physics. In *30th Conference on Neural Information Processing Systems* (NIPS 2016).
36. Li, Z. et al. Multipole graph neural operator for parametric partial differential equations. *Adv. Neural Inf. Process. Syst.* **33**, 6755–6766 (2020).
37. Kress, R., Maz'ya, V. & Kozlov, V. *Linear Integral Equations* Vol. 82 (Springer, 1989).
38. Greengard, L. & Rokhlin, V. A new version of the fast multipole method for the Laplace equation in three dimensions. *Acta Numer.* **6**, 229–269 (1997).
39. Trefethen, L. *Spectral Methods in MATLAB. Software, Environments, and Tools* (Society for Industrial and Applied Mathematics, 2000).
40. Kovachki, N., Lanthaler, S. & Mishra, S. On universal approximation and error bounds for Fourier neural operators. *J. Mach. Learn. Res.* **22**, 1–76 (2021).
41. Leonard, A. in *Turbulent Diffusion in Environmental Pollution. Advances in Geophysics* Vol. 18, 237–248 (Elsevier, 1975).
42. Temam, R. *Navier–Stokes Equations: Theory and Numerical Analysis* (Elsevier, 2016).
43. Chen, T. & Chen, H. Universal approximation to nonlinear operators by neural networks with arbitrary activation functions and its application to dynamical systems. *EEE Trans. Neural Netw.* **6**, 911–917 (1995).
44. Lingsch, L., Michelis, M., Perera, S. M., Katzschmann, R. K. & Mishra, S. A Structured matrix method for nonequispaced neural operators. Preprint at https://doi.org/10.48550/arXiv.2305.19663 (2023).
45. Li, Z., Huang, D. Z., Liu, B. & Anandkumar, A. Fourier neural operator with learned deformations for PDEs on general geometries. *J. Mach. Learn. Res.* **388**, 1–26 (2023).
46. Sun, H., Ross, Z. E., Zhu, W. & Azizzadenesheli, K. Next-generation seismic monitoring with neural operators. Preprint at https://doi.org/10.48550/arXiv.2305.03269 (2023).
47. Zou, C., Azizzadenesheli, K., Ross, Z. E. & Clayton, R. W. Deep neural Helmholtz operators for 3D elastic wave propagation and inversion. Preprint at https://doi.org/10.48550/arXiv.2311.09608 (2023).
48. Kossaifi, J., Kovachki, N. B., Azizzadenesheli, K. & Anandkumar, A. Multi-grid tensorized Fourier neural operator for high resolution PDEs. Preprint at https://doi.org/10.48550/arXiv.2310.00120 (2022).
49. Rahman, M. A., Ross, Z. E. & Azizzadenesheli, K. U-no: U-shaped neural operators. Preprint at https://arxiv.org/abs/2204.11127 (2023).
50. Raonić, B., Molinaro, R., Rohner, T., Mishra, S. & de Bezenac, E. Convolutional neural operators. *Advances in Neural Information Processing Systems 36* (NeurIPS) (2023).
51. Cao, S. Choose a transformer: Fourier or Galerkin. *Adv. Neural Inf. Process. Syst.* **34**, 24924–24940 (2021).
52. Li, Z., Meidani, K. & Farimani, A. B. Transformer for partial differential equations' operator learning. Preprint at https://arxiv.org/abs/2205.13671 (2023).
53. Lee, S. Mesh-independent operator learning for partial differential equations. In *ICML 2022 2nd AI for Science Workshop*. https://openreview.net/pdf?id=JUtZG8-2vGp (ICML, 2022).
54. Hao, Z. et al. GNOT: A general neural operator transformer for operator learning. In *Proc. 40th International Conference on Machine Learning* Vol. 202, 12556–12569, (PMLR, 2023).
55. Dosovitskiy, A. et al. An image is worth 16×16 words: transformers for image recognition at scale. In *Proc. 9th International Conference on Learning Representations* (ICLR, 2021).
56. Guibas, J. et al. Adaptive Fourier neural operators: efficient token mixers for transformers. In *Proc. 10th International Conference on Learning Representations* (ICLR, 2022).
57. Lagaris, I. E., Likas, A. & Fotiadis, D. I. Artificial neural networks for solving ordinary and partial differential equations. *IEEE Trans. Neural Netw.* **9**, 987–1000 (1998).
58. Karniadakis, G. E. et al. Physics-informed machine learning. *Nat. Rev. Phys.* **3**, 422–440 (2021).
59. Sirignano, J. & Spiliopoulos, K. Dgm: A deep learning algorithm for solving partial differential equations. *J. Comput. Phys.* **375**, 1339–1364 (2018).
60. Yu, B. et al. The deep ritz method: a deep learning-based numerical algorithm for solving variational problems. *Commun. Math. Stat.* **6**, 1–12 (2018).
61. Du, Y. & Zaki, T. A. Evolutional deep neural network. *Phys. Rev. E* **104**, 045303 (2021).
62. Mildenhall, B. et al. Nerf: representing scenes as neural radiance fields for view synthesis. *Commun. ACM* **65**, 99–106 (2021).
63. Sitzmann, V., Martel, J., Bergman, A., Lindell, D. & Wetzstein, G. Implicit neural representations with periodic activation functions. *Adv. Neural Inf. Process. Syst.* **33**, 7462–7473 (2020).
64. Jeong, Y. et al. PeRFception: perception using radiance fields. *Adv. Neural Inf. Process. Syst.* **35**, 26105–26121 (2022).
65. Srinivasan, P. P. et al. NeRV: neural reflectance and visibility fields for relighting and view synthesis. In *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 7495–7504 (IEEE, 2021).
66. Chen, P. Y. et al. CROM: continuous reduced-order modeling of PDEs using implicit neural representations. In *Proc. 11th International Conference on Learning Representations* (ICLR, 2023).
67. Serrano, L. et al. Operator learning with neural fields: tackling PDEs on general geometries. In *37th Conference on Neural Information Processing Systems* (NeurIPS 2023).
68. Fang, Z., Wang, S. & Perdikaris, P. Learning only on boundaries: a physics-informed neural operator for solving parametric partial differential equations in complex geometries. *Neural Comput.* **36**, 475–498 (2024).
69. Raissi, M., Perdikaris, P. & Karniadakis, G. E. Physics-informed neural networks: a deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *J. Comput. Phys.* **378**, 686–707 (2019).
70. Han, J., Jentzen, A. & E, W. Solving high-dimensional partial differential equations using deep learning. *Proc. Natl Acad. Sci. USA* **115**, 8505–8510 (2018).
71. Smith, J. D., Azizzadenesheli, K. & Ross, Z. E. EikoNet: solving the Eikonal equation with deep neural networks. *IEEE Trans. Geosci. Remote Sens.* **59**, 10685–10696 (2020).
72. Gao, H., Sun, L. & Wang, J.-X. PhyGeoNet: physics-informed geometry-adaptive convolutional neural networks for solving parameterized steady-state PDEs on irregular domain. *J. Comput. Phys.* **428**, 110079 (2021).
73. Krishnapriyan, A., Gholami, A., Zhe, S., Kirby, R. & Mahoney, M. W. Characterizing possible failure modes in physics-informed neural networks. *Adv. Neural Inf. Process. Syst.* **34**, 26548–26560 (2021).
74. Wang, S., Wang, H. & Perdikaris, P. Learning the solution operator of parametric partial differential equations with physics-informed DeepONets. *Sci. Adv.* **7**, eabi8605 (2021).
75. Goswami, S., Bora, A., Yu, Y. & Karniadakis, G. E. *Physics-Informed Deep Neural Operator Networks* (Springer,2023); https://doi.org/10.1007/978-3-031-36644-4_6 (2023).
76. Berner, J., Dablander, M. & Grohs, P. Numerically solving parametric families of high-dimensional Kolmogorov partial differential equations via deep learning. *Adv. Neural Inf. Process. Syst.* **33**, 16615–16627 (2020).
77. Han, J., Nica, M. & Stinchcombe, A. R. A derivative-free method for solving elliptic partial differential equations with deep neural networks. *J. Comput. Phys.* **419**, 109672 (2020).
78. Beck, C., Becker, S., Grohs, P., Jaafari, N. & Jentzen, A. Solving the Kolmogorov PDE by means of deep learning. *J. Sci. Comput.* **88**, 1–28 (2021).
79. Richter, L. & Berner, J. Robust SDE-based variational formulations for solving linear PDEs via deep learning. In *International Conference on Machine Learning*, 18649–18666 (PMLR, 2022).
80. Zhang, R. et al. Monte Carlo neural operator for learning PDEs via probabilistic representation. Preprint at https://doi.org/10.48550/arXiv.2302.05104 (2023).
81. Rahman, M. A., Florez, M. A., Anandkumar, A., Ross, Z. E. & Azizzadenesheli, K. Generative adversarial neural operators. Preprint at https://arxiv.org/abs/2205.03017 (2022).
82. Lim, J. H. et al. Score-based diffusion models in function space. Preprint at https://doi.org/10.48550/arXiv.2302.07400 (2023).
83. Seidman, J. H., Kissas, G., Pappas, G. J. & Perdikaris, P. Variational autoencoding neural operators. *Proc. 40th International Conference on Machine Learning* Vol. 202, 30491–30522 (PMLR, 2023).
84. Shi, Y., Lavrentiadis, G., Asimaki, D., Ross, Z. E. & Azizzadenesheli, K. Broadband ground motion synthesis via generative adversarial neural operators: development and validation. Preprint at https://doi.org/10.48550/arXiv.2309.03447 (2023).
85. Lam, R. et al. Learning skillful medium-range global weather forecasting. *Science* **382**, 1416–1421 (2023).
86. How AI models are transforming weather forecasting: a showcase of data-driven systems. *ECMWF* www.ecmwf.int/en/about/media-centre/news/2023/how-ai-models-are-transforming-weather-forecasting-showcase-data (6 September 2023).
87. Grady, T. J. et al. Model-parallel Fourier neural operators as learned surrogates for large-scale parametric PDEs. *Comput. Geosci.* **178**, 105402 (2023).
88. Renn, P. I. et al. Forecasting subcritical cylinder wakes with Fourier neural operators. Preprint at https://doi.org/10.48550/arXiv.2301.08290 (2023).
89. Li, Z., Peng, W., Yuan, Z. & Wang, J. Fourier neural operator approach to large eddy simulation of three-dimensional turbulence. *Theor. Appl. Mech. Lett.* **12**, 100389 (2022).
90. Peng, W. et al. Fourier neural operator for real-time simulation of 3D dynamic urban microclimate. Preprint at https://doi.org/10.48550/arXiv.2308.03985 (2023).
91. Liu, B. et al. A learning-based multiscale method and its application to inelastic impact problems. *J. Mech. Phys. Solids.* **158**, 104668 (2022).
92. Rashid, M. M., Pittie, T., Chakraborty, S. & Krishnan, N. A. Learning the stress-strain fields in digital composites using Fourier neural operator. *iScience* **25** 105452 (2022).
93. Liu, M. et al. An adversarial active sampling-based data augmentation framework for manufacturable chip design. In *36th Conference on Neural Information Processing Systems* (NeurIPS 2022); https://doi.org/10.48550/arXiv.2210.15765.
94. Yang, H. et al. Generic lithography modeling with dual-band optics-inspired neural networks. In *Proc. 59th ACM/IEEE Design Automation Conference* 973–978 (IEEE, 2022).
95. Guan, S., Hsu, K.-T. & Chitnis, P. V. Fourier neural operator networks: a fast and general solver for the photoacoustic wave equation. *Algorithms* **16**, 124 (2023).
96. Gu, J. et al. Neurolight: a physics-agnostic neural operator enabling parametric photonic device simulation. *Adv. Neural Inf. Process. Syst.* **35**, 14623–14636 (2022).
97. Gopakumar, V. et al. Fourier neural operator for plasma modelling. *Nuclear Fission* https://doi.org/10.1088/1741-4326/ad313a (2024).
98. Li, Z. et al. Learning chaotic dynamics in dissipative systems. *Adv. Neural Inf. Process. Syst.* **35**, 16768–16781 (2022).
99. Lippe, P., Veeling, B. S., Perdikaris, P., Turner, R. E. & Brandstetter, J. PDE-refiner: achieving accurate long rollouts with neural PDE solvers. In *37th Conference on Neural Information Processing Systems* (NeurIPS 2023).
100. Liu-Schiaffini, M. et al. Tipping point forecasting in non-stationary dynamics on function spaces. Preprint at https://doi.org/10.48550/arXiv.2308.08794 (2023).

101. Rosen, P. A., Gurrola, E., Sacco, G. F. & Zebker, H. The InSAR scientific computing environment. In *EUSAR 2012; 9th European Conference on Synthetic Aperture Radar* 730–733 (VDE, 2012).
102. Palmer, T. Stochastic weather and climate models. *Nat. Rev. Phys.* **1**, 463–471 (2019).
103. Salvi, C., Lemercier, M. & Gerasimovics, A. Neural stochastic pdes: Resolution-invariant learning of continuous spatiotemporal dynamics. *Adv. Neural Inf. Process. Syst.* **35**, 1333–1344 (2022).
104. Ngom, M. & Marin, O. Fourier neural networks as function approximators and differential equation solvers. *Statist. Anal. Data Mining* **14**, 647–661 (2021).
105. Shi, Y. et al. Machine learning accelerated PDE backstepping observers. In *2022 IEEE 61st Conference on Decision and Control (CDC)* 5423–5428 (IEEE, 2022).
106. Cotter, S., Roberts, G., Stuart, A. & White, D. MCMC methods for functions: modifying old algorithms to make them faster. *Statist. Sci.* **28**, 424–446 (2012).
107. Hinze, M., Pinnau, R., Ulbrich, M. & Ulbrich, S. *Optimization with PDE Constraints. Mathematical Modelling: Theory and Applications* (Springer, 2008).
108. Kaltenbach, S., Perdikaris, P. & Koutsourelakis, P.-S. Semi-supervised invertible neural operators for Bayesian inverse problems. *Computational Mechanics* 1–20 (2023).
109. Zhou, T. et al. AI-aided geometric design of anti-infection catheters. *Sci. Adv.* **10**, adj1741 (2024).
110. Yang, Y. et al. Seismic wave propagation and inversion with neural operators. *Seism. Rec.* **1**, 126–134 (2021).
111. Sun, H., Yang, Y., Azizzadenesheli, K., Clayton, R. W. & Ross, Z. E. Accelerating time-reversal imaging with neural operators for real-time earthquake locations. Preprint at https://doi.org/10.48550/arXiv.2210.06636 (2022).
112. Yang, Y., Gao, A. F., Azizzadenesheli, K., Clayton, R. W. & Ross, Z. E. Rapid seismic waveform modeling and inversion with neural operators. *IEEE Trans. Geosci. Remote Sens.* https://doi.org/10.1109/TGRS.2023.3264210 (2023).
113. Yin, Z., Orozco, R., Louboutin, M. & Herrmann, F. J. Solving multiphysics-based inverse problems with learned surrogates and constraints. *Adv. Model. Simul. Eng. Sci.* **10**, 14 (2023).
114. Otness, K. et al. An extensible benchmark suite for learning to simulate physical systems. In *35th Conference on Neural Information Processing Systems* (NeurIPS 2021).
115. Takamoto, M. et al. PDEbench: an extensive benchmark for scientific machine learning. *Adv. Neural Inf. Process. Syst.* **35**, 1596–1611 (2022).
116. Gupta, J. K. & Brandstetter, J. Towards multi-spatiotemporal-scale generalized PDE modeling. Preprint at https://doi.org/10.48550/arXiv.2209.15616 (2022).
117. Hao, Z. et al. Pinnacle: a comprehensive benchmark of physics-informed neural networks for solving PDEs. Preprint at https://doi.org/10.48550/arXiv.2306.08827 (2023).
118. Huang, Z. et al. *A Large-Scale Benchmark for the Incompressible Navier–Stokes Equations* (SSRN, 2022); https://doi.org/10.2139/ssrn.4030476.
119. Ren, P. et al. Superbench: a super-resolution benchmark dataset for scientific machine learning. Preprint at https://doi.org/10.48550/arXiv.2306.14070 (2023).
120. Wang, Z., Bovik, A. C., Sheikh, H. R. & Simoncelli, E. P. Image quality assessment: from error visibility to structural similarity. *IEEE Trans. Image Process.* **13**, 600–612 (2004).
121. Hassan, S. M. S. et al. BubbleML: a multi-physics dataset and benchmarks for machine learning. Preprint at https://doi.org/10.48550/arXiv.2307.14623 (2023).
122. Dulny, A., Hotho, A. & Krause, A. *Dynabench: A Benchmark Dataset for Learning Dynamical Systems From Low-Resolution Data* Vol. 14169 (Springer, 2023); https://doi.org/10.1007/978-3-031-43412-9_26.
123. Thiyagalingam, J., Shankar, M., Fox, G. & Hey, T. Scientific machine learning benchmarks. *Nat. Rev. Phys.* **4**, 413–420 (2022).
124. Kurth, T. et al. FourCastNet: accelerating global high-resolution weather forecasting using adaptive Fourier neural operators. In *PASC '23: Proc. Platform for Advanced Scientific Computing Conference* https://doi.org/10.1145/3592979.3593412 (ACM, 2023).
125. White, C. et al. Speeding up Fourier neural operators via mixed precision. Preprint at https://doi.org/10.48550/arXiv.2307.15034 (2023).

Acknowledgements
A.A. is supported by a Bren named professor chair at Caltech and AI 2050 senior fellowship by Schmidt Sciences. Z.L. is supported by an NVIDIA fellowship. M.L.-S. is supported by the Mellon Mays undergraduate fellowship. We thank B. Jenik for creating Fig. 2 and for general discussions.

Author contributions
The authors contributed equally to all aspects of the article.

Competing interests
The authors declare no competing interests.

Additional information
Peer review information *Nature Reviews Physics* thanks Cristopher Salvi and the other, anonymous, reviewer(s) for their contribution to the peer review of this work.