Journal of Computational Physics 519 (2024) 113379

Contents lists available at ScienceDirect

Journal of Computational Physics

journal homepage: www.elsevier.com/locate/jcp

PDE generalization of in-context operator networks: A study on 1D scalar nonlinear conservation laws

Liu Yang, Stanley J. Osher *

Department of Mathematics, University of California, Los Angeles, 520 Portola Plaza, Los Angeles, 90095, CA, USA

ARTICLE INFO

Dataset link: https://github.com/LiuYangMage/in-context-operator-networks

Keywords:
In-context learning
Operator learning
Conservation laws
Machine learning

ABSTRACT

Can we build a single large model for a wide range of PDE-related scientific learning tasks? Can this model generalize to new PDEs without any fine-tuning? In-context operator learning and the corresponding model In-Context Operator Networks (ICON) represent an initial exploration of these questions. The capability of ICON regarding the first question has been demonstrated previously. In this paper, we present a detailed methodology for solving PDE problems with ICON, and show how a single ICON model can make forward and reverse predictions for different equations with different strides, provided with appropriately designed data prompts. We show positive evidence for the second question above, through a study on 1D scalar nonlinear conservation laws, a family of PDEs with temporal evolution. In particular, we show that an ICON model trained on conservation laws with cubic flux functions can generalize well to some other flux functions of more general forms, without fine-tuning. We also show how to broaden the range of problems that an ICON model can address, by transforming functions and equations to ICON's capability scope. We believe that the progress in this paper is a significant step towards the goal of training a foundation model for PDE-related tasks under the in-context operator learning framework.

1. Introduction

Looking back to the evolution of neural network solvers for partial differential equations (PDEs), we see a three-act progression.

Act 1 focuses on approximating the solution functions with neural networks. Typical (but not exhaustive) examples include [1] which represents an early exploration to solve PDEs with neural networks, [2] and [3] for high-dimensional parabolic PDEs and backward stochastic differential equations, Deep Galerkin Method (DGM) [4], Deep Ritz Method (DRM) [5], Physics-Informed Neural Networks (PINNs) [6], Weak Adversarial Network (WAN) [7] to impose PDEs of different forms in the training loss function, [8,9] to solve high-dimensional optimal transport and mean-field control/game problems, and APAC-Net [10] for mean-field game problems with the primal-dual formulation.

Despite their success, in Act 1 the neural network needs to be trained again when the solution function changes due to changes in the equation or the initial/boundary conditions. Such limitation leads to Act 2, where the neural networks are employed to approximate solution operators, namely "operator learning". Here an operator transforms one or multiple input functions, termed as the "condition" in this paper, to an output function, termed as the "quantity of interest (QoI)" in this paper. A wide range of

* Corresponding author.
E-mail address: sjo@math.ucla.edu (S.J. Osher).

https://doi.org/10.1016/j.jcp.2024.113379
Received 16 January 2024; Received in revised form 8 July 2024; Accepted 26 August 2024

Available online 29 August 2024

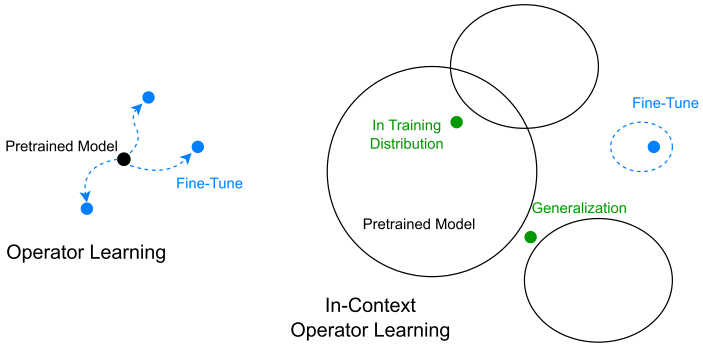

Fig. 1. Operator learning v.s. in-context operator learning. For classic operator learning, one model is limited to approximate a single operator, with the need of fine-tuning when the operator changes to a "close" new one. For in-context operator learning, a single model can approximate a wide range of operators. It can serve as a "foundation model" that could be directly applied without fine-tuning for different PDE-related tasks in the training distribution, or even beyond due to generalization in the operator space. It can also be fine-tuned to strengthen its expertise in particular operator domains, if necessary.

scientific machine learning tasks can be formulated as operator learning problems. Take the task of solving time-independent PDEs for instance, the condition could be the coefficient function, with QoI being the solution, and different equations corresponding to different operators. For optimal control problems, the condition could correspond to the initial state, while the QoI embodies the control signal, with different control dynamics corresponding to different operators.

Operator learning can be traced back to [11,12] where shallow neural networks are used to approximate nonlinear operators. More recent examples include (but are not limited to) [13] for parametric PDE problems, [14] for problems governed by stochastic PDEs, PDE-Net [15] for PDEs with temporal evolution, Fourier Neural Operator (FNO) [16,17] with integral kernel in Fourier space to learn the solution operator, Deep Operator Network (DeepONet) [18,19] which maps the parameters or the initial/boundary conditions to the solutions, [20] and Physics-Informed Neural Operators (PINO) [21] for parametric PDEs, [22–24] with graph neural networks. Other related work includes [25–29].

The above methods have successfully demonstrated the capability of neural networks in approximating solution operators. However, in these methods, one model is limited to approximating a single operator, and thus needs to be retrained when the operator changes, which could be due to even a minor change in the PDE.

While the cost of retraining can be reduced by fine-tuning a pretrained model in Act 1 [30–39] and Act 2 [21,27,40,28,29,41–43], we remark that the model must be fine-tuned individually for each distinct function or operator. When there are substantial changes to the target function/operator, such fine-tuning may be expensive or even fail. These drawbacks significantly limit the applicability of the model.

In-context operator learning [44] can be viewed as the initial attempt of Act 3, where a single model can manage multiple operators. Other examples in Act 3 include ICON-LM [45] with text and data prompts, PROSE [46] which can generate both the symbolic expression of the governing system and the operator network for multiple distinct ODEs, [47] for multiple ODEs with in-context operator learning, and Multiple Physics Pretraining (MPP) [48] where a single model learns different physics from different history records.

In-context operator learning approach draws inspiration from in-context learning for Large Language Models (LLMs) [49,50], where the model performs a task specified by the prompted "context", including task descriptions and a few related examples. Recently, the in-context learning paradigm has been applied to other domains, including function regression [51], large vision models [52], language-vision models [53], robotics/embodied models [54], etc. We refer the readers to the survey [55] for more details on recent advances in this topic.

For in-context operator learning, the corresponding model "in-context operator network (ICON)" acts as an "operator learner" rather than being tuned to approximate a specific operator. The trained ICON model is able to infer the operator from the prompted condition-QoI example pairs, and apply the inferred operator to new question conditions for corresponding QoI. Such a learning process happens during the forward pass, without the need for weight adjustments. With different prompted examples, a single ICON model approximates different operators, and thus can tackle a wide range of scientific learning tasks.

For instance, in [44], a single ICON model adeptly manages 19 distinct equations/types of operators, encompassing forward and inverse ordinary differential equations (ODEs), PDEs, and mean-field control problems, with each type containing infinitely many operators and condition/QoI functions being 1D or 2D. In [45], ICON is further evolved to take multi-modal prompts, including data examples and texts that describe the task. The demonstrated efficacy hints at the potential for training a "foundation model" under the in-context operator learning framework. Such a model could be applied directly for a wide range of PDE-related tasks or, if necessary, be fine-tuned to strengthen its expertise in particular operator domains. We show the comparison between operator learning and in-context operator learning in Fig. 1.

Large Language Models have demonstrated impressive generalization capabilities, even beyond human expectations [56]. In [44], ICON also demonstrated generalization to operators that are not included in the training distribution. However, the generalization is limited to equations with out-of-distribution parameters, and the generalization to new forms of equations is not observed there. Also, [47] showed the generalization of in-context operator learning in ODEs.

This paper aims to (1) present a detailed methodology for PDE forward and reversal predictions under the in-context operator learning framework, and (2) explore the generalization capabilities of ICON for PDEs.

In particular, we focus on conservation laws, a family of PDEs with temporal evolution. Our investigation encompasses both time forward and reverse operators. The forward operator takes the initial state as the condition and predicts the system's future state as the QoI, while the reverse operator is conceptualized by making time-reverse predictions, i.e., inverting the roles of the condition and the QoI.

We are motivated to study conservation laws by the following reasons.

1. Conservation laws are foundational in describing a wide array of real-world systems.
2. The family of conservation laws is rich, making it ideal for testing the generalization capabilities of ICON.
3. The complexities inherent in conservation laws present interesting challenges, including the discontinuities in the solution function, as well as the non-uniqueness of the time-reverse solution.

In this paper, we show that an ICON model trained on conservation laws with cubic flux functions can generalize well to some other flux functions of more general forms, without fine-tuning. We believe that this is a significant step towards the goal of training a foundation model for PDE-related tasks under the in-context operator learning framework.

2. Method

2.1. In-context operator network (ICON)

In this paper, we employ the ICON-LM model introduced in [45], which is an improved variant of the ICON model introduced in [44], where "LM" stands for "language model". ICON-LM enables multi-modal learning, taking texts and numerical data as input, but it also supports single-modal learning, i.e. learning with numerical data without texts. For single-modal learning, compared with the vanilla ICON model, ICON-LM model exhibits improved training efficiency, achieving better accuracy with about half of the parameters and less training time. In this study, we focus on single-modal learning of ICON-LM model, and for simplicity, we refer to it as "ICON".

Denote the neural network as $\mathcal{T}_\theta$ with parameters θ. ICON takes a sequence of condition-QoI pairs as input, and predicts each QoI through one forward pass, i.e.

$$\{\text{prediction of QOI}_i\}_{i=2}^{I} = \mathcal{T}_\theta[\{\langle \text{COND}_i, \text{QOI}_i\rangle\}_{i=1}^{I}]. \tag{1}$$

Here COND_i denotes i-th condition, and QOI_i denotes i-th QoI. Note that $\{\langle \text{COND}_i, \text{QOI}_i\rangle\}_{i=1}^{I}$ should be associated with a shared operator, and the operator could be different across different sequences. Each condition or QoI function is represented by a set of key-value pairs, each key-value pair representing a token in the input sequence of the ICON model.

With a transformer-based architecture and a special attention mask, the prediction of $J+1$-th QoI relies only on the previous condition-QoI pairs (from the first pair to the J-th pair), as well as $J+1$-th condition itself.[1] This can be written as:

$$\text{prediction of QOI}_{J+1} = \mathcal{T}_\theta[\text{COND}_{J+1}; \{\langle \text{COND}_i, \text{QOI}_i\rangle\}_{i=1}^{J}], J = 1, 2, \ldots I-1. \tag{2}$$

We don't make predictions for QOI_1, since it makes no sense to predict it just with COND_1 without any condition-QoI pairs as examples to indicate the operator.

This approach, termed "next function prediction", mirrors the "next token prediction" approach in language models. The model is trained by comparing the predicted QoIs and the ground truth QoIs in the sequence, e.g., using the mean squared error as the loss function.

After training, through one forward pass, the ICON model can ingest examples of condition-QoI pairs, learn the corresponding operator, apply it to the question condition, and predict the corresponding QoI. We only need to view the question condition as the last one in the sequence. This relationship can be formulated as:

$$\text{prediction of QOI}_{\text{question}} = \mathcal{T}_\theta[\text{COND}_{\text{question}}; \{\langle \text{COND}_i, \text{QOI}_i\rangle\}_{i=1}^{J}], \tag{3}$$

where J is the number of examples used in the prompt for in-context operator learning. A single ICON model can handle different J, ranging from one to the maximum capacity $I-1$.

We remark these condition-QoI pairs in training and inference may not necessarily come from classical numerical solvers. Sometimes they are derived from experiments or real-world sensors, where exact PDEs or initial/boundary conditions may not be known. Such scenarios present challenges for traditional numerical methods, making them more suitable for data-driven approaches.

For more details of the ICON model, readers are directed to [45].

[1] Technically, the input sequence consists of condition-QoI-query tuples. These queries are just the inputs of the query functions, indicating where to evaluate the predicted QoI functions. They are conceptually auxiliary, less important than conditions and QoIs, and could be dropped in future variants of ICON. We thus omit them in our notations for simplicity.

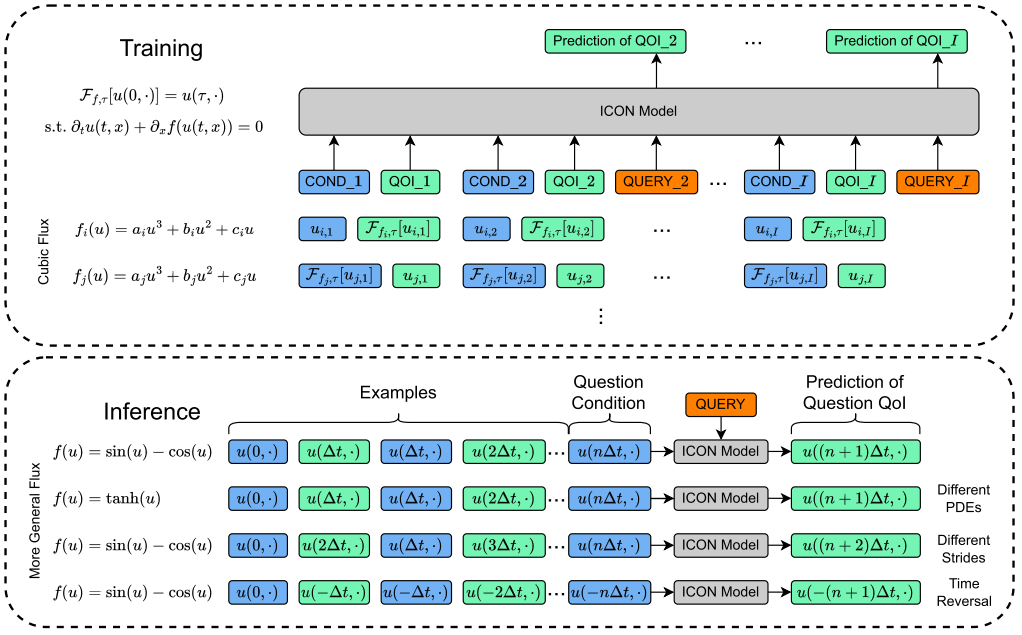

Fig. 2. Illustration of training and inference of ICON for PDEs, using conservation laws as examples.

2.2. Forward and reverse operators

For PDEs with temporal evolution, the forward operator takes the initial state as the condition and the state at time τ later as the QoI, while the condition and QoI are swapped in reverse operators. Earlier in [44] ICON already demonstrated the capability of in-context operator learning for forward operators in control problems. In this paper, we focus on 1D scalar nonlinear conservation laws that take the following form:

$$\partial_t u(t,x) + \partial_x f(u(t,x)) = 0, x \in [0,1], \tag{4}$$

with periodic boundary condition, where u is the solution state, f is the flux. Here we fix the geometry and boundary conditions, so that the forward and reverse operators are defined by the flux f and temporal intervals (or "stride" for simplicity) τ. In particular, the forward operator $\mathcal{F}_{f,\tau}$ is defined as

$$\begin{aligned} &\mathcal{F}_{f,\tau}[u(0,\cdot)] = u(\tau,\cdot) \\ \text{s.t.} \quad &\partial_t u(t,x) + \partial_x f(u(t,x)) = 0 \end{aligned} \tag{5}$$

The reverse operator is defined as

$$\mathcal{R}_{f,\tau}[u] = \{v | \mathcal{F}_{f,\tau}[v] = u\} \tag{6}$$

Note that the reverse solution is not unique, thus we use the set to include all feasible solutions.

Each condition or QoI function $u(t,\cdot)$ is represented by a set of key-value pairs. In this paper, the key is the spatial coordinate x, and the value is the average value of the solution u over the interval $[x, x+\Delta x]$ with $\Delta x = 0.01$, for $x = 0, 0.01, ..., 0.99$, i.e. 100 key-value pairs or tokens for each function.

2.3. Training

To build the training data, we can simulate different PDEs with different initial conditions and obtain the condition-QoI pairs from the simulation records. Note that one record can be used to generate multiple condition-QoI pairs.

In particular, in this paper, the training PDEs are 1D scalar conservation laws with cubic flux functions, i.e.,

$$\partial_t u + \partial_x (au^3 + bu^2 + cu) = 0, \tag{7}$$

with a, b, c uniformly sampled from $[-1, 1]$. We fix the forward and reverse stride $\tau = 0.1$ during training, i.e., the training forward and reverse operators are $\mathcal{F}_{au^3+bu^2+cu,0.1}$ and $\mathcal{R}_{au^3+bu^2+cu,0.1}$ respectively, with a, b, c uniformly sampled from $[-1, 1]$. The training process is illustrated in Fig. 2.

Recall that the ICON model takes a sequence of condition-QoI pairs as input, and predicts each QoI based on the previous condition-QoI pairs as well as the current condition. It is straightforward to train the model for forward operators: the training loss is defined as the mean squared error between the predicted QoI and the ground truth QoI, i.e. the L_2 loss:

$$L_{\text{Forward}}(\theta) = \frac{1}{I-1}\sum_{i=2}^{I} \|\text{prediction of QOI}_i - \text{QOI}_i\|^2 \tag{8}$$

The reverse operators are more subtle since the solution is not unique. In this paper, we discuss two options:

1. L_2 **Loss**: The reverse L_2 loss is the same as in Equation (8). The only difference is that the condition and QoI are swapped.
2. **Consistency Loss**: If we use the model as a surrogate of the exact forward operator, and apply it to the predicted QoI, e.g. $u(0,\cdot)$, ideally we should recover the condition, e.g. $u(\tau,\cdot)$. We employ the surrogate forward operator since the exact forward simulation is too expensive in the training loss function. The consistency loss can be built based on the recovered condition and the ground truth. More formally,

$$\begin{aligned} L_{\text{Consistency}}(\theta) &= \frac{1}{I-1}\sum_{i=2}^{I} \|\hat{\mathcal{F}}_i[\text{prediction of QOI}_i] - \text{COND}_i\|^2 \\ \text{where } \hat{\mathcal{F}}_i[\cdot] &= \mathcal{T}_\theta[\cdot; \{\langle \text{QOI}_j, \text{COND}_j\rangle\}_{j=1,j\neq i}^{I}] \end{aligned} \tag{9}$$

   Here we use $\{\langle \text{QOI}_j, \text{COND}_j\rangle\}_{j=1,j\neq i}^{I}$, i.e., all but i-th condition-QoI pairs with the condition and QoI swapped, as examples to indicate the forward operator. The parameters θ are frozen in $\hat{\mathcal{F}}_i$, i.e., the flow of gradients is blocked, to reduce the computational cost.

Mathematically, the reverse L_2 loss aims to find the expectation of multiple feasible solutions, which strictly speaking may not be a feasible solution. The consistency loss is more rigorous but also computationally more expensive. Moreover, the effectiveness of the consistency loss heavily depends on the accuracy of the forward operator surrogate. We compare the two options in Section 4.2.

In the end, we remark that we didn't use automatic differentiation in the loss function to incorporate the PDE information. Instead, ICON learns from data alone, in particular, generated by the third-order Weighted Essentially Non-Oscillatory (WENO) scheme [57] in this paper. This strategy not only reduces the training costs but also significantly enhances the robustness of the training process, especially when dealing with discontinuities. Indeed, the battle-tested stability and accuracy of the numerical scheme play a critical role in handling the discontinuities in the solutions of conservation laws. The details of data generation are presented in Section 3.

2.4. Inference

After training, a single ICON model can be applied to various PDEs or condition functions. During such an inference phase, there are no weight updates at all, thus the computational cost is extremely small. The model's adaptability comes from the way we construct the data prompts, including the example condition-QoI pairs and the question condition, similar to designing prompts for a language model.

We emphasize that the PDE is not explicitly fed into the model via parameters or other means. Instead, the model learns the PDE implicitly from the condition-QoI examples. Therefore, the model can naturally be applied to PDEs with new forms, as long as the condition-QoI examples are constructed correspondingly. In this paper, we train the model with cubic flux functions, and test the model with flux functions of more general forms. Also, while the time stride τ is fixed in training, it can vary in inference, since this is equivalent to scaling the PDE properly, as we will discuss in Section 2.6.2. The inference phase is illustrated in Fig. 2.

Based on the source of the condition-QoI examples, there are three cases:

1. **Self Reference**: In this case, for data we have a sequence

$$(u(0), u(\Delta t), u(2\Delta t), \cdots, u(n\Delta t)) =: (u(i\Delta t))_{i=0}^{n}, \tag{10}$$

   where $u(t)$ is the solution of the PDE at time t. We call the sequence a "record", and each $u(t)$ a "frame". We will construct condition-QoI examples from the given record, and predict future frames, i.e. $u((n+1)\Delta t), u((n+2)\Delta t), \cdots$, or previous frames, i.e. $u(-\Delta t), u(-2\Delta t), \cdots$.
2. **Single Reference Record**: In this case, we have a reference record and make predictions for a new initial/terminal condition.
3. **Multiple Reference Records**: In this case, we have multiple reference records governed by the same PDE, and make forward/reverse predictions for a given initial/terminal condition.

"Self Reference" can be viewed as a special case of "Single Reference Record", but we list it separately since it is a very common case in practice. For all these cases, we can construct a condition-QoI example by randomly sampling a frame from a record as the condition, and a later/previous frame as the QoI from the same record with a certain stride.

After the introduction of ICON, other researchers also looked into training one model for multi-physics prediction, by specifying physics with history records [48]. In particular, in the inference stage, the model input is the sequence $(u(i\Delta t))_{i=0}^{n}$ which implicitly

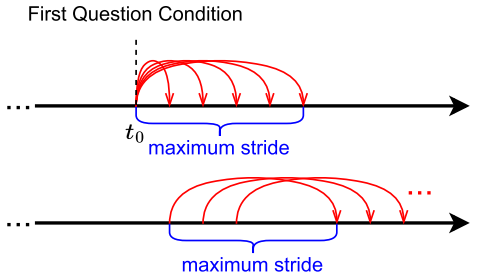

Fig. 3. Illustration of the scheme for recursive predictions.

encodes the physics, and the output is the prediction of $u((n+1)\Delta t)$. The model is used in an auto-regressive manner to make predictions for $u((n+2)\Delta t), u((n+3)\Delta t), \cdots$. We refer to this approach as "video prediction" since it draws a parallel between forecasting PDE and predicting video frames.

While this approach is capable of addressing multiple PDEs, it overlooks an important property: common PDEs are time-homogeneous Markovian processes, i.e. the future state only depends on the current state, and such dependence is invariant in time.[2] This property is not only a key feature of most PDEs, but also forms the foundation of both forward and reverse operator definitions, as well as classic numerical PDE schemes.

By taking advantage of the time-homogeneous Markovian property of PDEs, and constructing condition-QoI examples in a very flexible way, ICON is more powerful.

If we construct the condition-QoI examples as

$$\langle u(0), u(\Delta t)\rangle, \langle u(\Delta t), u(2\Delta t)\rangle, \cdots, \langle u((n-1)\Delta t), u(n\Delta t)\rangle, \tag{11}$$

and set $u(n\Delta t)$ as the question condition, then the ICON model will recover the video prediction approach and predict $u((n+1)\Delta t)$. Here we use the notation $\langle u(t_1), u(t_2)\rangle$ to denote a condition-QoI example, where $u(t_1)$ is the condition and $u(t_2)$ is the QoI.

Beyond one-step prediction, ICON can easily make multi-step or large-stride predictions by simply changing the order of frames in the input sequence. For example, suppose n is a multiple of 2 and we want to make predictions for $u((n+n/2)\Delta t)$, we can construct condition-QoI examples as

$$\langle u(0), u(n/2\Delta t)\rangle, \langle u(\Delta t), u((n/2+1)\Delta t)\rangle, \cdots, \langle u(n/2\Delta t), u(n\Delta t)\rangle, \tag{12}$$

and set $u(n\Delta t)$ as the question condition, then the model will predict $u((n+n/2)\Delta t)$ with one forward pass.

To make reverse predictions, given the sequence $(u(-i\Delta t))_{i=0}^{n}$, we can construct condition-QoI examples as

$$\langle u(0), u(-\Delta t)\rangle, \langle u(-\Delta t), u(-2\Delta t)\rangle, \cdots, \langle u(-(n-1)\Delta t), u(-n\Delta t)\rangle, \tag{13}$$

and set $u(-n\Delta t)$ as the question condition, so the ICON model will learn the reverse operator and predict $u(-(n+1)\Delta t)$. Similarly for multi-step reverse predictions.

In the end, we remark that ICON is adaptive to non-Markovian processes as well: we just need to set the condition as multiple frames instead of one.

2.5. Recursive predictions

To make predictions for a long time horizon, we can recursively call the ICON model with the predicted QoIs as the new question conditions. Since a single ICON model can make predictions with different strides, there are multiple schemes to recursively make predictions. We present one example here, which is used throughout the paper.

Given the record(s) for building condition-QoI examples, as well as u_{t_0} as the first question condition, we make recursive forward predictions with the following steps:

1. Prescribe the maximum stride S.
2. Using $u(t_0)$ as the question condition, make predictions of $u(t_0+s)$ with the stride s, for $s=\Delta t, 2\Delta t, ..., S$.
3. Using $u(t-S)$ as the question condition, make predictions of $u(t)$ with the maximum stride S, for $t=t_0+S+\Delta t, t_0+S+2\Delta t, \cdots$.

An illustration of the recursive prediction scheme is shown in Fig. 3. It's clear that we can make predictions for $t_0+n\Delta t$ for $n\in\mathbb{N}$ in this way. The recursive reverse predictions can be made similarly.

[2] Some terms in the PDE can be time-dependent, but usually these terms are the system states or controls, and the PDE that governs the system states or controls is time-homogeneous Markovian.

2.6. Transform operators and functions to ICON's capability scope

ICON can handle a range of operators and condition functions through in-context operator learning. When faced with unfamiliar operators and/or condition functions, it's sometimes possible to transform them into ones that ICON can effectively process. Such techniques broaden the range of problems that ICON can address and strengthen its role as a foundation model. As an analogy, the concept of "chain of thought" [58] has been introduced to decompose complex reasoning tasks into smaller, more manageable segments that can be tackled by a Large Language Model.

Here, we demonstrate two simple examples of such techniques: change of variables and varying strides.

2.6.1. Change of variables

When applying the change of variables, both the equation and condition function are transformed. Consider the conservation law $\partial_t u + \partial_x f(u) = 0$, take a simple affine transformation as example, i.e., $u = \alpha v + \beta$ with $\alpha > 0$, then the equation becomes

$$\partial_t v + \partial_x f(\alpha v + \beta)/\alpha = 0, \tag{14}$$

In other words, the condition function is transformed to $v = (u - \beta)/\alpha$, and the flux function is transformed to $f(\alpha v + \beta)/\alpha$.

For condition functions that are beyond the training scale, we can apply the affine transformation with $\alpha > 1$, so that it can be scaled down to the training scale. However, we remark that α cannot be too large. Firstly, the prediction error will be amplified by α when transforming back to the original variables. Secondly, the flux $f(\alpha v + \beta)/\alpha$ could go out of the training distribution for a large α. The detailed numerical results are presented in Section 4.5.

The training PDEs are limited to conservation laws with cubic flux functions in this paper, and we thus only considered linear transform. If the training distribution covers more general cases, more sophisticated transformations can also be effective.

2.6.2. Varying strides

For conservation laws of form (4), we can see that

$$\mathcal{F}_{kf,\tau} = \mathcal{F}_{f,k\tau}, \quad \mathcal{R}_{kf,\tau} = \mathcal{R}_{f,k\tau}, \tag{15}$$

where k is a positive constant. In other words, if we apply a smaller/larger stride, it is equivalent to solving a PDE with a smaller/larger flux and the original stride. This technique can be used to adjust the scale of flux functions to the desired range.

Note that with a smaller stride, it takes more steps to reach the same time, i.e., more calls to the ICON model. This could lead to a larger error accumulation. Therefore we need to make a tradeoff between the generalization ability of ICON and the error accumulation. The detailed numerical results are presented in Section 4.6.

3. Data preparation

The training data are generated by simulating conservation laws with cubic flux functions:

$$\partial_t u + \partial_x(au^3 + bu^2 + cu) = 0, \quad x \in [0, 1], \tag{16}$$

with periodic boundary conditions. The simulation is conducted with the third-order WENO scheme [57], a battle-tested numerical scheme for conservation laws.

For the training phase, we employ a set of 1000 tuples of (a, b, c), each randomly sampled from the hypercube $[-1, 1]^3$. The following protocol is adopted for each tuple of operator parameters (a, b, c):

1. **Initial Conditions:** Sample $N = 100$ periodic functions as initial conditions. These functions are defined on a grid with spacing $\Delta x = 0.01$. In practice, we sample these functions from a periodic Gaussian random field with zero mean and covariance kernel
$$k(x, x') = \sigma^2 \exp\left(-\frac{1 - \cos(2\pi(x - x'))}{l^2}\right), \tag{17}$$
where $\sigma = 1, l = 1$. The initial functions with value beyond $[-3, 3]$ are dropped for the sake of the CFL condition during data generation.
2. **Numerical Solution:** Employ the third-order WENO scheme and the fourth-order Runge-Kutta method to solve the conservation law. The system evolves from $t = 0$ to $t = 0.5$ using a time step $\Delta t = 0.0005$, resulting in 1001 steps in total, taking account of both the initial and final states. The solution we get is the weak solution which satisfies the jump condition for the discontinuities, and also satisfies the entropy condition.
3. **Data Collection:** Consider each of the first 801 time steps corresponding to the initial 0.4 time units as an individual initial condition. Each of these has a corresponding function that occurs 0.1 time units later. Such pairs of functions can be collected as the conditions and QoIs for the forward and reverse operators.

Given N initial functions, this procedure yields a total of $801N = 80100$ conditions-QoIs pairs for each operator. To optimize storage efficiency, we do not exhaustively utilize all generated condition-QoI pairs. Instead, for each operator we randomly downsample and store $100N = 10000$ pairs for training.

Table 1
Transformer configuration.

Layers	6
Heads in Multi-Head Attention	8
Input/Output Dimension of Each Layer	256
Dimension of Query/Key/Value in Attention Function	256
Hidden Dimension of Feedforward Networks	1024

Table 2
Configuration of optimizer and learning rate schedule.

Initial Learning Rate	0.0
Peak Learning Rate	1e-4
End Learning Rate	0.0
Training Steps	10^6
Warmup Steps	First 10% of Total Steps
Cosine Annealing Steps	Remaining Steps
Global Norm Clip	1.0
Adam β_1	0.9
Adam β_2	0.999
Adam Weight Decay	1e-4

We used JAX for efficient data preparation, completing the training data generation in approximately 1.6 hours using a single NVIDIA RTX 4090 GPU.

4. Experimental results

The ICON model employs a transformer architecture, in this paper configured as in Table 1. For optimization, the AdamW optimizer is used in conjunction with a warmup-cosine-decay schedule, following the parameters set in Table 2.

The input and output layers are linear layers. In this paper, the input sequence during training consists of six condition-QoI pairs. Consequently, the inference phase is limited to a maximum of five examples, plus a single question condition. The condition/QoI functions and the five additional queries related to QoIs (excluding the first one) comprise 100 tokens each, as is explained in Section 2.2. This results in a total of 1700 tokens in the input sequence during training.

4.1. Metrics

Denote the trained neural network as $\mathcal{T}_\theta$ as in Equation (1), then given some examples $\{\langle u_i; \mathcal{F}_{f,\tau}[u_i]\rangle\}_i^I$, we can approximate the forward operator $\mathcal{F}_{f,\tau}$ with $\mathcal{T}_\theta[\cdot; \{\langle u_i, \mathcal{F}_{f,\tau}[u_i]\rangle\}_i^I]$, and the reverse operator $\mathcal{R}_{f,\tau}$ with $\mathcal{T}_\theta[\cdot; \{\langle \mathcal{F}_{f,\tau}[u_i], u_i\rangle\}_i^I]$.

We denote

$$\begin{aligned} \hat{\mathcal{F}}_{f,\tau}[u] &:= \mathcal{T}_\theta[u; \{\langle u_i, \mathcal{F}_{f,\tau}[u_i]\rangle\}_i^I], \\ \hat{\mathcal{R}}_{f,\tau}[u] &:= \mathcal{T}_\theta[u; \{\langle \mathcal{F}_{f,\tau}[u_i], u_i\rangle\}_i^I]. \end{aligned} \tag{18}$$

The condition-QoI examples $\{\langle u_i, \mathcal{F}_{f,\tau}[u_i]\rangle\}_i^I$ and $\{\langle \mathcal{F}_{f,\tau}[u_i], u_i\rangle\}_i^I$ are dropped for simplicity. We will specify them as needed in the following sections.

To quantify the performance of these learned operators, we define two metrics: the forward error and the reverse error. The forward error is straightforward, which is the average L_1 distance between the predicted QoI and the ground truth QoI:

$$\text{Forward Error} := \|\hat{\mathcal{F}}_{f,\tau}[u] - \mathcal{F}_{f,\tau}[u]\|_1, \tag{19}$$

The reverse error is more subtle since the solution is not unique. We thus apply the exact forward operator[3] to the predicted QoI, and compare the result with the condition function:

$$\text{Reverse Error} := \|\mathcal{F}_{f,\tau}[\hat{\mathcal{R}}_{f,\tau}[u]] - u\|_1. \tag{20}$$

Note that here we use the exact forward operator $\mathcal{F}_{f,\tau}$ in Reverse Error, as oppose to the surrogate model $\hat{\mathcal{F}}_{f,\tau}$ in $L_{\text{Consistency}}$ for training. This is because we want to focus on evaluating the reverse prediction and avoid the error in the forward operator approximation.

For recursive forward predictions, the error is also defined as the average L_1 distance between the predictions and the ground truth. For recursive reverse predictions, we apply the exact forward operator to the predictions until t_0, the time for the first question condition, and calculate the L_1 distance between the reconstructed first question condition and the ground truth.

[3] Here we use forward simulation to apply the "exact" forward operator. Strictly speaking, such an operator also has numerical errors and thus is not "exact". However, the errors are negligible compared with neural network prediction errors, and we thus ignore them.

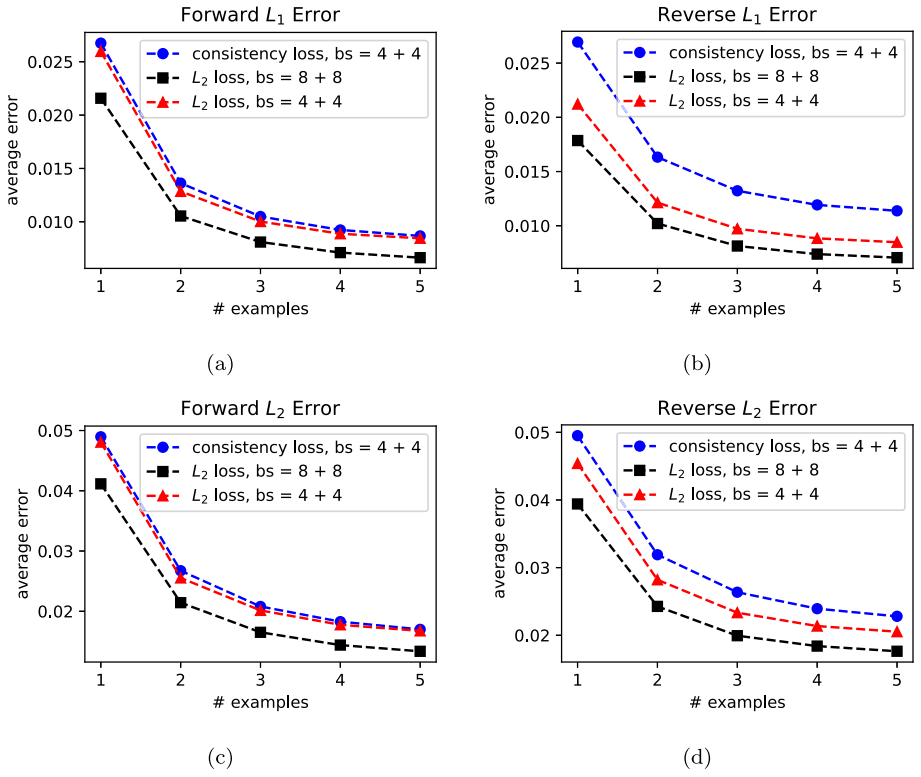

Fig. 4. Averaged error v.s. the number of examples used for in-context operator learning. (a) Forward L_1 error. (b) Reverse L_1 error. (c) Forward L_2 error. (d) Reverse L_2 error.

Here we use the L_1 distance as the evaluation metric, a common choice for analyzing conservation laws with discontinuities. This is different from the L_2 loss function used in training. One can also define the L_2 error by replacing $\|\cdot\|_1$ with $\|\cdot\|_2$ in Equation (19) and (20). We made a comparison between L_1 error and L_2 error in Section 4.2.

4.2. *In-distribution operators*

In this section, we evaluate the performance of ICON on in-distribution operators, specifically focusing on the forward operators $\mathcal{F}_{au^3+bu^2+cu,0.1}$ and reverse operators $\mathcal{R}_{au^3+bu^2+cu,0.1}$, with the number of examples ranging from 1 to 5. Here (a,b,c) are on the $11\times 11\times 11$ uniform grid within $[-1,1]^3$. For each operator, the condition-QoI pairs are generated in the same way as for training.

We also compared the L_2 loss and consistency loss for training reverse operators. In particular, we consider three training configurations:

1. L_2 loss with batch size 4 for forward operators and consistency loss with batch size 4 for reverse operators. The training takes about 45.5 hours on dual NVIDIA RTX 4090 GPUs.
2. L_2 loss with expected batch size 8 for forward operators and another 8 for reverse operators.[4] The training takes about 40 hours on dual NVIDIA RTX 4090 GPUs.
3. L_2 loss with expected batch size 4 for forward operators and another 4 for reverse operators. The training takes about 37 hours on a single NVIDIA RTX 4090 GPU.

In Fig. 4, we show the forward error and reverse error averaged over the operators on the grid and 100 instances of in-context operator learning for each operator. For all three configurations, it's clear that both errors decay as the number of in-context examples increases. The consistency loss performs the worst in our experiments, although with the largest computational cost. This may be due to that the model cannot serve as an accurate forward operator surrogate before convergence, and thus the consistency loss is not effective. A more sophisticated training strategy can be developed to improve the performance of consistency loss, e.g., gradually increasing the weight of consistency loss as the model converges. Since this is not the focus of this paper, we leave it as future work.

[4] When using L_2 loss for both forward and reverse operators, practically we train the model with data randomly sampled from forward and reverse operators. The exact split in each iteration may be different, but the expectation is half-half. When using consistency loss, in each iteration, we use the same 4 sequences to build the forward L_2 loss and the consistency loss.

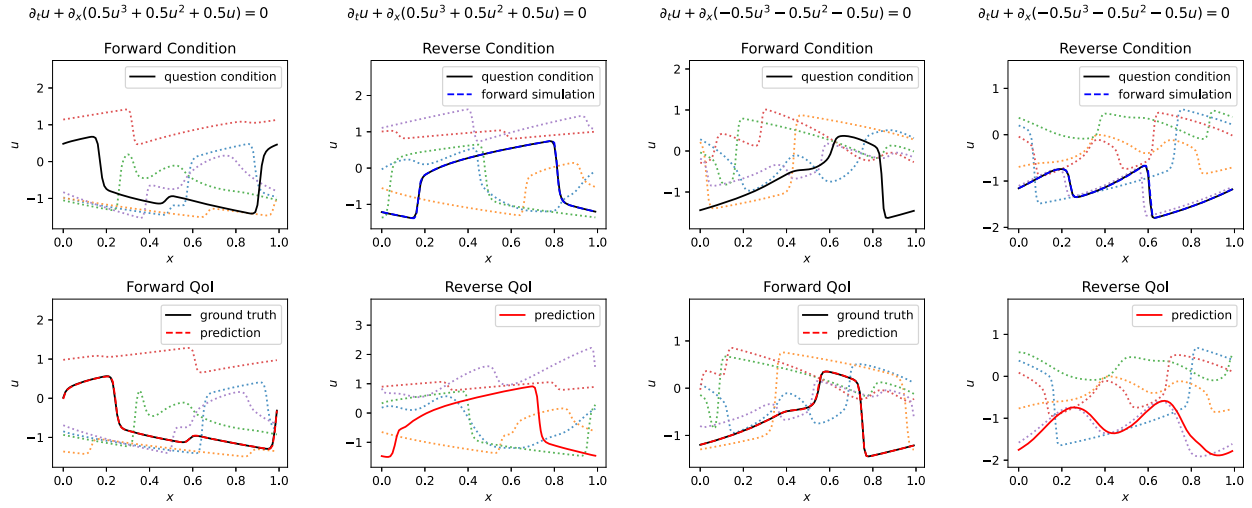

Fig. 5. Illustration of in-context operator learning for $\mathcal{F}_{0.5u^3+0.5u^2+0.5u,0.1}$, $\mathcal{R}_{0.5u^3+0.5u^2+0.5u,0.1}$, $\mathcal{F}_{-0.5u^3-0.5u^2-0.5u,0.1}$ and $\mathcal{R}_{-0.5u^3-0.5u^2-0.5u,0.1}$. For each case, the prompted five condition-QoI examples are shown with dotted color lines. The forward predictions shown with dashed red lines overlap with the ground truth QoIs shown with solid black lines. Since there are no unique ground truth solutions for the reverse operators, we apply the exact forward operators to the predicted QoIs by forward simulation, and show the recovered conditions with dashed blue lines, which overlap with the question conditions shown with solid black lines. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

In Fig. 4, the L_2 error is approximately twice the L_1 error; however, the qualitative conclusions are consistent. Therefore, we will continue to use the L_1 distance as the evaluation metrics in subsequent sections.

Since the L_2 loss with $8+8$ batch size works the best, from now on, we will analyze the results using the model trained with this configuration. As an illustration, in Fig. 5 we show some cases of in-context forward and reverse operator learning, corresponding to different equations. The overlapping between the predicted QoI and the ground truth QoI for forward operators, and the overlapping between the recovered condition and the question condition for reverse operators, indicate that the learned operators are accurate.

4.3. Comparison with classic operator learning

In this section, we compare ICON with classic operator learning methods, including FNO [16] and DeepONet [18].

In this study, we utilize a DeepONet model with approximately 15 million parameters and an FNO model with about 20 million parameters, comparable to the ICON model with about 16 million parameters. Classic operator learning methods typically train models to approximate a specific operator. Here, we initially pretrain both the DeepONet and FNO models using approximately 1.5 GB of data to approximate $\mathcal{F}_{0.2u^3+0.2u^2+0.2u,0.1}$. Subsequently, these pretrained models are fine-tuned individually to approximate other operators, including $\mathcal{F}_{0.21u^3+0.21u^2+0.21u,0.1}$, $\mathcal{F}_{0.25u^3+0.25u^2+0.25u,0.1}$, and $\mathcal{F}_{0.30u^3+0.30u^2+0.30u,0.1}$. The fine-tuning process involves 1000 steps using the AdamW optimizer, featuring a constant learning rate 1e-5, weight decay 1e-4, and global norm clip 1.0. We use a batch size that matches the total number of available examples, ranging from 5 to 1000.

As illustrated in Fig. 6, the pretrained FNO model exhibits superior accuracy in approximating $\mathcal{F}_{0.2u^3+0.2u^2+0.2u,0.1}$ compared to ICON, whereas DeepONet shows slightly poorer performance. Without fine-tuning, the models are unable to predict other operators accurately. For fine-tuning, adding more examples incrementally enhances accuracy, with operators further from the initial training target requiring more examples. For example, fine-tuning FNO with 5 examples suffices for approximating $\mathcal{F}_{0.21u^3+0.21u^2+0.21u,0.1}$ with an accuracy comparable to that of ICON, whereas approximately 300 examples are required for $\mathcal{F}_{0.3u^3+0.3u^2+0.3u,0.1}$.

Furthermore, we report the runtimes of different methods in Table 3. These runtimes were measured on a single NVIDIA RTX 4090 GPU, with batch sizes set to one for WENO simulation and inference of models. A WENO simulation of 200 steps with a step size of 0.0005 corresponds to a duration of 0.1, identical to the time stride used for ICON and classic operator learning methods. ICON's inference process is faster than WENO simulation. The inference times for FNO and DeepONet are quicker than for ICON, however, their fine-tuning processes are substantially slower. For different setups, it takes different steps for the fine-tuning processes to converge, but even one fine-tuning step with batch size 5 is slower than the inference of ICON. As a reference of fine-tuning steps, in Fig. 7, we show the error versus the number of steps during the fine-tuning of FNO to approximate $\mathcal{F}_{0.3u^3+0.3u^2+0.3u,0.1}$, where approximately 100 to several hundred steps are required for convergence.

We acknowledge that the hyperparameters of the operator learning models, such as depth and width, were not meticulously optimized. Adjustments to these parameters might enhance the performance of DeepONet and FNO. Also, different implementations might accelerate the training and inference. However, the need for fine-tuning models for individual operators and the considerable computational expense of fine-tuning compared to a single forward pass in the ICON inference likely remain unchanged.

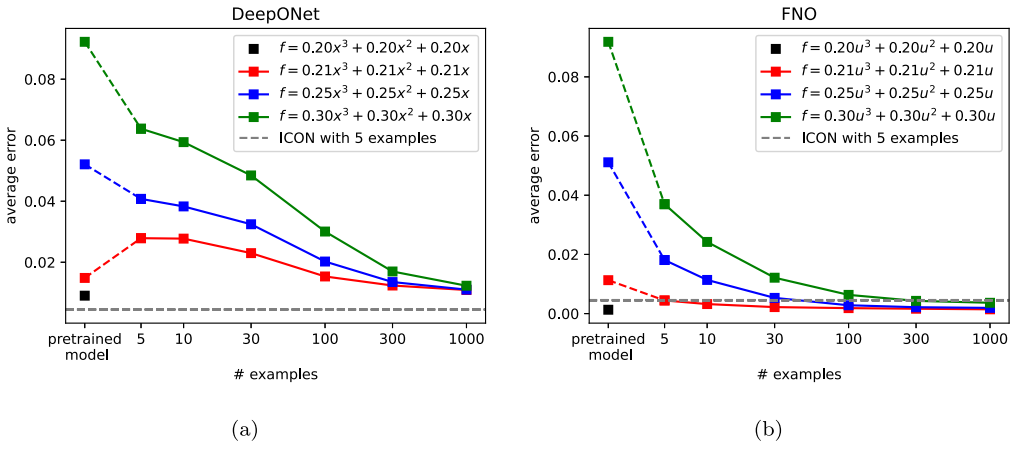

Fig. 6. Accuracy comparison between ICON and classic operator learning methods, with errors averaged across 100 independent groups of data for fine-tuning and testing. (a) Error of DeepONet (b) Error of FNO. The errors for ICON across the four operators are represented by grey dashed lines, which are closely aligned and virtually indistinguishable.

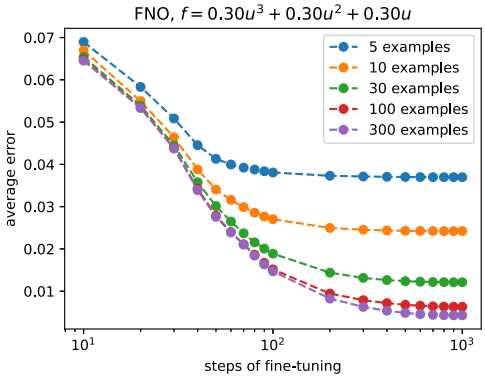

Fig. 7. Error versus the number of steps during the fine-tuning of FNO to approximate $\mathcal{F}_{0.3u^3+0.3u^2+0.3u,0.1}$, with errors averaged across 100 independent groups of data for fine-tuning and testing. Here the batch size matches the total number of available examples.

Table 3
Approximate running time of different methods.

Method	Operation	Time (ms)
WENO	Simulation (200 steps)	16.4
ICON	Inference (5 examples in the prompt)	3.5
DeepONet	Inference	0.8
	Fine-tuning per step (batch size = 5)	4.0
FNO	Inference	2.0
	Fine-tuning per step (batch size = 5)	7.4

4.4. Generalization to new PDEs

In this section, we show that the ICON can generalize to new PDEs with more general forms of flux functions.

We apply ICON recursively to make predictions for a long time horizon. In particular, for forward predictions, we consider that we have 11 frames of data $u(0), u(0.01), \cdots, u(0.1)$, and use the trained ICON to predict $u(0.11), u(0.12) \cdots, u(0.5)$. We follow the recursive scheme introduced in Section 2.5, with $t_0 = 0.1$, $\Delta t = 0.01$, and the maximum stride $S = 0.05$. For all predictions, we use five condition-QoI pairs as prompted examples, which are randomly sampled from

$$\langle u(0), u(s)\rangle, \langle u(0.01), u(0.01+s)\rangle, \cdots, \langle u(0.1-s), u(0.1)\rangle, \tag{21}$$

when making predictions with a stride of s.

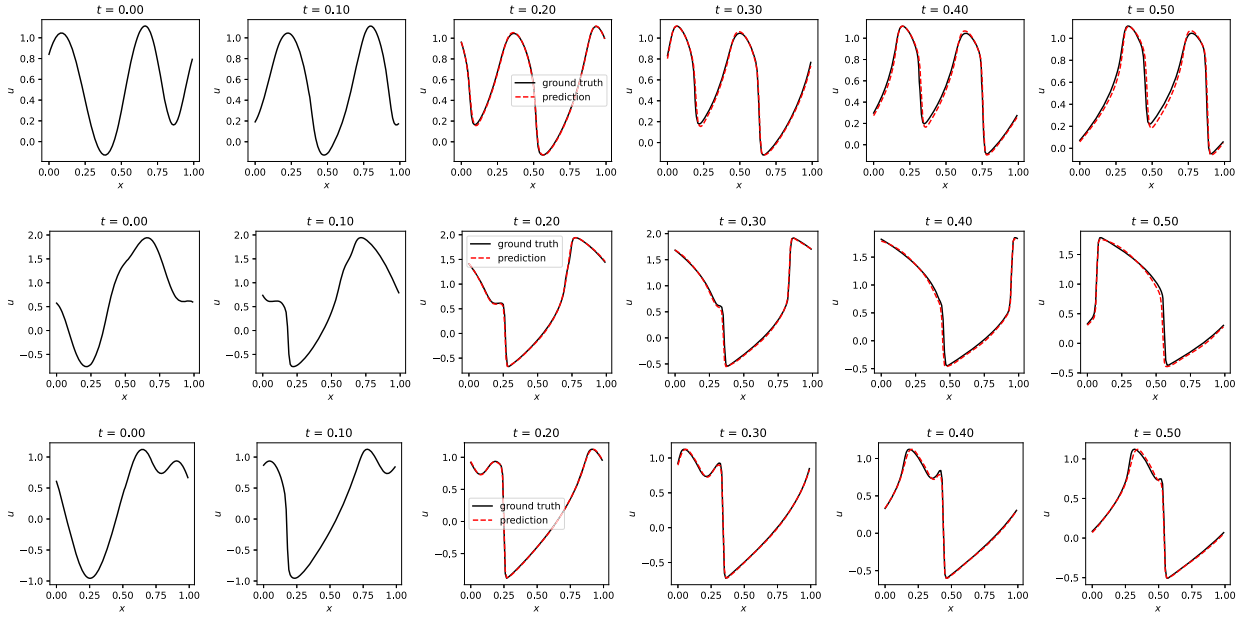

Fig. 8. Three examples of forward prediction of $\partial_t u + \partial_x(\sin(u) - \cos(u)) = 0$. The overlapping between the forward predictions and the ground truth data indicates that the forward predictions are accurate.

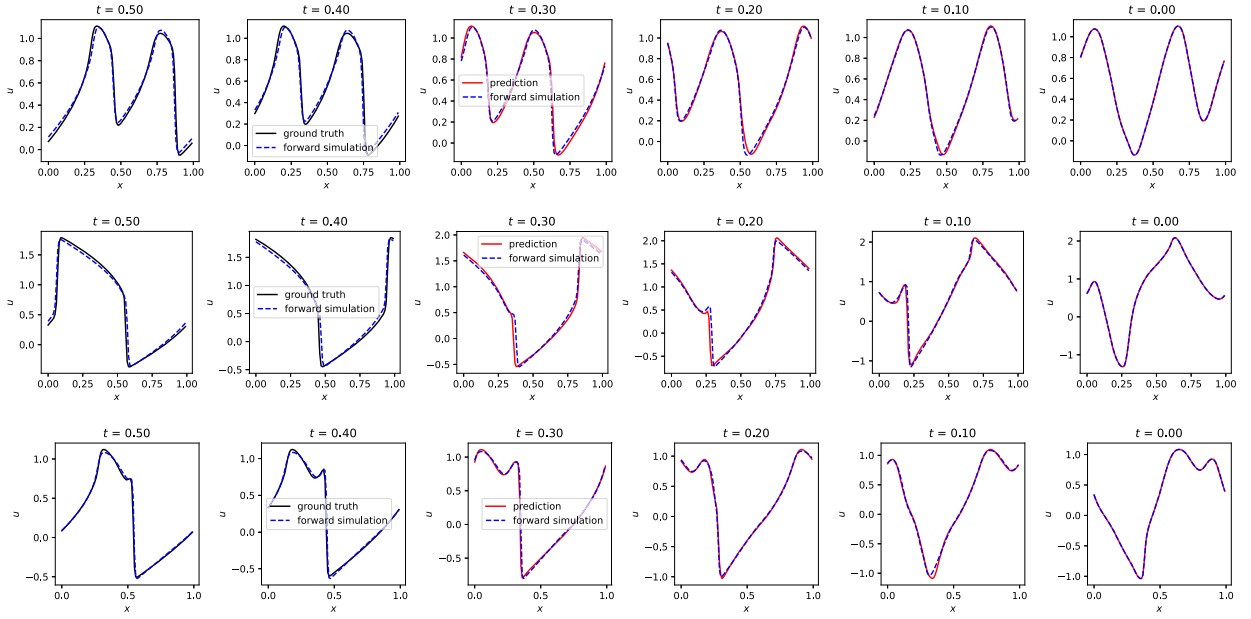

Fig. 9. Three examples of reverse prediction of $\partial_t u + \partial_x(\sin(u) - \cos(u)) = 0$. We apply the exact forward simulation from the predicted initial condition at $t = 0$ until $t = 0.5$, shown with blue dashed lines. The overlapping between the forward simulation results and ground truth data indicates that the reverse predictions are accurate.

Similarly, for reverse predictions, we consider that we have 11 frames of data $u(0.5), u(0.49), \cdots, u(0.4)$, and use the trained ICON to predict $u(0.39), u(0.38) \cdots, u(0)$, with $t_0 = 0.4$.

In Fig. 8 and 9 we showcase some forward and reverse prediction results for $\partial_t u + \partial_x(\sin(u) - \cos(u)) = 0$. We can see the great accuracy, even for the PDE and initial functions that are never seen during training.

How accurate is the prediction for the new PDE compared with the predictions for PDEs with cubic flux functions? Did the ICON model simply memorize the cubic flux functions and approximate the new flux with the closest cubic function?

To answer these questions, we consider the following two comparisons:

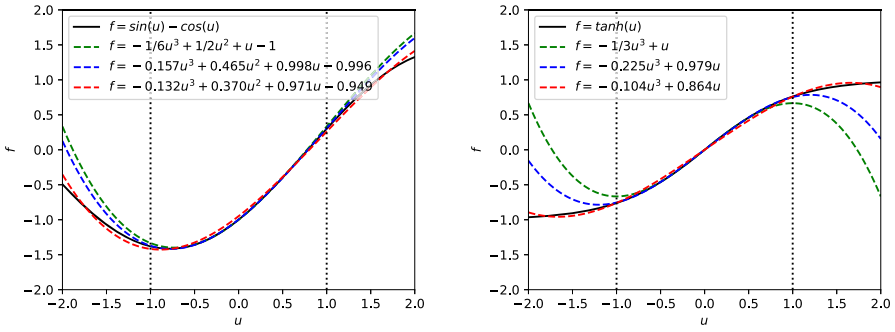

Fig. 10. The tested flux functions (black solid lines) and "similar" cubic functions (colored dashed lines), including cubic Taylor polynomials (green), cubic fit in $[-1,1]$ (blue), and cubic fit in $[-2,2]$ (red). (a) $f=\sin(u)-\cos(u)$. (b) $f=\tanh(u)$.

Comparison 1: We compare the errors of predictions for different equations, including $f(u)=\sin(u)-\cos(u)$ and "similar" cubic flux functions. The definition of "similar" will be introduced later.
Comparison 2: We use different equations, including $f(u)=\sin(u)-\cos(u)$ and "similar" cubic flux functions, to generate the prompted examples. We make predictions with these examples, and compare the errors between the predictions and the ground truth corresponding to $f=\sin(u)-\cos(u)$. To ensure a fair comparison, the initial condition for generating prompted examples via simulation, as well the first question condition $u(t_0)$ for recursive predictions, are shared across all equations.

The "similar" cubic flux functions consist of the following:

1. The cubic Taylor polynomial of $f(u)=\sin(u)-\cos(u)$ at 0, i.e. $f(u)=-1/6u^3+1/2u^2+u$. The constant term is dropped since it does not affect the solution. Same for the following.
2. The best cubic fit of $f(u)=\sin(u)-\cos(u)$ within $[-1,1]$ in the L_2 sense, i.e., $f(u)=-0.157u^3+0.465u^2+0.998u$ approximately.
3. The best cubic fit of $f(u)=\sin(u)-\cos(u)$ within $[-2,2]$ in the L_2 sense, i.e., $f(u)=-0.132u^3+0.370u^2+0.971u$ approximately.
4. Since the range of u varies for PDE with different initial conditions, we also consider the best cubic fit of $f(u)=\sin(u)-\cos(u)$ within $[u_{\min},u_{\max}]$ in the L_2 sense, where $u_{\min}$ and $u_{\max}$ are the minimum and maximum values of u in the initial condition. Due to the maximum principle, the range of u will not exceed $[u_{\min},u_{\max}]$ for all time. We denote this as "adaptive cubic fit".

The first three "similar" cubic functions are illustrated in Fig. 10a.

In Fig. 11 we show the errors of forward and reverse predictions w.r.t. time. These errors are averaged over 512 instances, with the initial conditions sampled from the same stochastic process as in training (with different random seeds).

There are two key observations:

1. For Comparison 1, the error for the new equation is higher than those for "similar" cubic flux functions, but still within a reasonable range. This is expected since the new equation is out of the training distribution.
2. For Comparison 2, when the prompted examples are generated from "similar" cubic flux functions, the errors are higher than that with "correct" examples. This shows that the ICON model didn't simply memorize the cubic flux functions and approximate the new flux with the closest cubic function (at least not in a trivial way). Instead, it is able to generalize to more general forms of flux functions.

In Fig. 12, we also show the results for $f(u)=\tanh(u)$, with "similar" cubic functions illustrated in Fig. 10b. The observations are consistent with those for $f(u)=\sin(u)-\cos(u)$.

4.5. Change of variables

In this section, we study the change of variables, focusing on the example equation $\partial_t u+\partial_x(\sin(u)-\cos(u))=0$.

We apply the affine transformation $v=(u-\beta)/\alpha$, so that $\partial_t v+\partial_x f(\alpha v+\beta)/\alpha=0$. Let $\beta=(u_{\max}+u_{\min})/2$, $\alpha=(u_{\max}-u_{\min})/(2r)$, so that if $u\in[u_{\min},u_{\max}]$, then $v\in[-r,r]$. Here we set $u_{\min}$ and $u_{\max}$ as the minimum and maximum values of u in the condition-QoI examples as well as the question condition. We feed v to the ICON model to make predictions, and then apply the inverse transformation $u=\alpha v+\beta$ to recover the predictions in the original variable u. The other setups are the same as in Section 4.4.

The errors for forward and reverse predictions are shown in Fig. 13. We can see that when $r=1$, i.e. scaling u to $[-1,1]$, the errors are the lowest. Recall the initial condition is sampled from the Gaussian process with mean 0 and variance 1, it is reasonable that $r=1$ works better than "no change of variables", or $r=2,3$. The errors for $r=0.5$ are higher, which can be attributed to two reasons discussed in Section 2.6.1: (1) the prediction errors are amplified when transforming back to the original variables, (2) the new flux $\sin(\alpha v+\beta)/\alpha-\cos(\alpha v+\beta)/\alpha$ could be far away from the distribution of cubic flux functions during training.

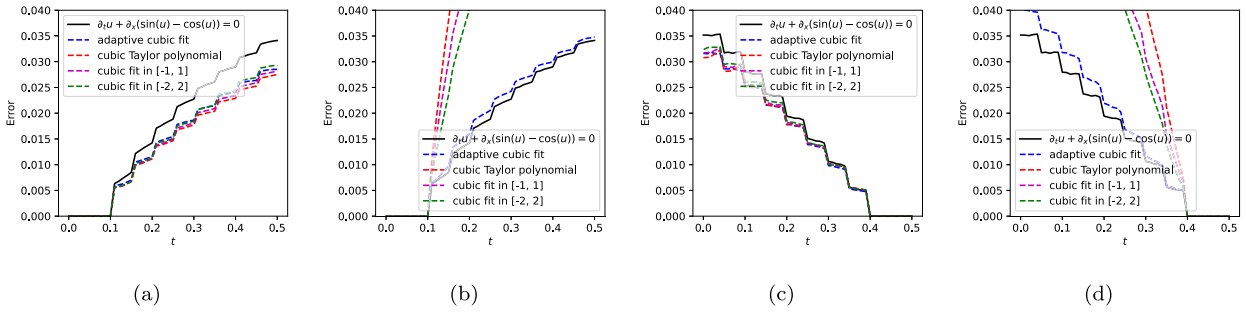

Fig. 11. Generalization for $\partial_t u + \partial_x(\sin(u) - \cos(u)) = 0$. (a) the error of forward prediction for different equations. (b) The error of forward prediction with prompted examples coming from different equations. (c) the error of reverse prediction for different equations. (d) The error of reverse prediction with prompted examples coming from different equations.

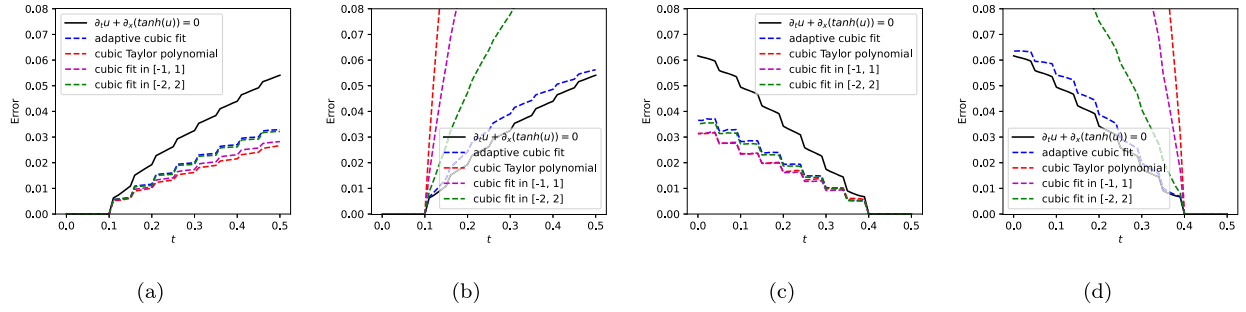

Fig. 12. Generalization for $\partial_t u + \partial_x(\tanh(u)) = 0$. The meanings of subfigures are the same as in Fig. 11.

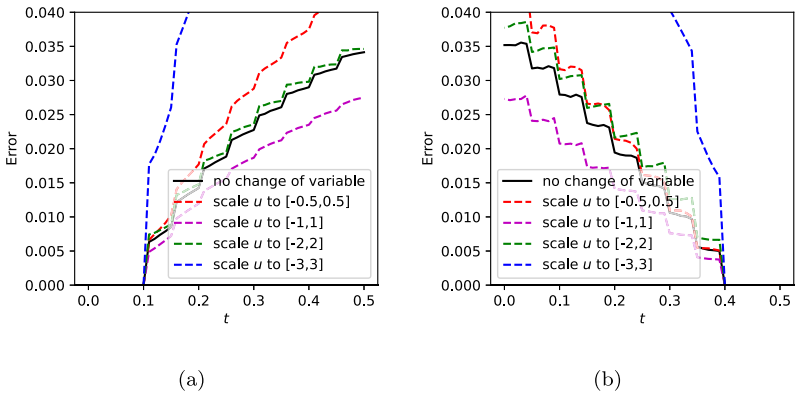

Fig. 13. The error of prediction for $\partial_t u + \partial_x(\sin(u) - \cos(u)) = 0$, with and without change of variables. (a) the error of forward prediction. (b) the error of reverse prediction. The error is the lowest when scaling u to $[-1, 1]$.

4.6. Varying strides

In this section, we study the effect of varying maximum stride S for recursive predictions. We consider the equation $\partial_t u + \partial_x(\sin(u) - \cos(u)) = 0$, and $\partial_t u + \partial_x(3\sin(u) - 3\cos(u)) = 0$, and make predictions with $S = 0.01, 0.02, ..., 0.05$. The other setups are the same as in Section 4.4. The results are shown in Fig. 14.

For $f = \sin(u) - \cos(u)$, the error increases as the maximum stride decreases from 0.05 to 0.01. The forward operators $\mathcal{F}_{\sin(u)-\cos(u),0.1k}$ are equivalent to $\mathcal{F}_{k\sin(u)-k\cos(u),0.1}$ for $k = 0.5, 0.4, ..., 0.1$. Since $\sin(u) - \cos(u) \approx -1/6u^3 + 1/2u^2 + u + const$, these operators are in the training range where the cubic polynomial coefficients are within $[-1, 1]^3$. The difference in the performance can be attributed to the error accumulation in recursive predictions, with a larger maximum stride leading to a smaller error due to less recursive steps.

For $f = 3\sin(u) - 3\cos(u)$, the forward operators $\mathcal{F}_{3\sin(u)-3\cos(u),0.1k}$ are equivalent to $\mathcal{F}_{3k\sin(u)-3k\cos(u),0.1}$ for $k = 0.5, 0.4, ..., 0.1$. One can see that the operators are in the training range for $k = 0.1, 0.2, 0.3$, while out of the training range for $k = 0.4, 0.5$. Therefore, the

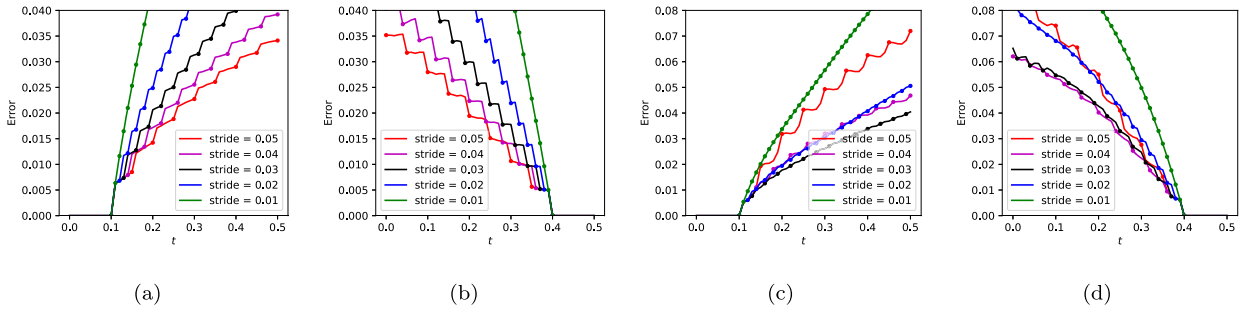

Fig. 14. The error of prediction with different maximum strides. (a, b) the error of prediction for $\partial_t u + \partial_x(\sin(u) - \cos(u)) = 0$. (c, d) the error of prediction for $\partial_t u + \partial_x(3\sin(u) - 3\cos(u)) = 0$. (a, c) the error of forward prediction. (b, d) the error of reverse prediction.

error decreases as the maximum stride increases from 0.01 to 0.03 due to error accumulation, and then increases as the maximum stride continues to increase from 0.03 to 0.05 due to the out-of-distribution effect.

5. Limitation and future directions

We have previously demonstrated the capabilities of ICON. In this section, we address its limitations and propose future research directions.

First, the generalization ability of ICON in this study is restricted to flux functions and time strides. Our preliminary findings show that ICON, when trained under periodic boundary conditions, fails to adapt to non-periodic conditions and struggles with prediction strides significantly larger than those in the training set. Enhancing generalization could be achievable by training a larger model on a broader dataset with diverse operators and initial functions. Future studies will explore whether ICON can exhibit emergent capabilities in large-scale models, similar to those observed in LLMs.

Second, the current recursive schemes with varying strides and the change of variables remain relatively simple. We are interested in developing more sophisticated and effective algorithms or models that can automate these processes, akin to the advancements in prompt engineering seen in LLMs, e.g., the chain of thought [58]. This exploration may open up a new research field that is built on top of foundation models for scientific computing.

Third, the efficiency and scalability of training and inference processes need to be enhanced. Currently, ICON relies on transformers, which incur a computational cost that scales quadratically with the number of tokens. This scalability issue becomes a significant barrier in high-dimensional applications, which require a large number of data points (possibly exponential to the dimension) to represent the functions. Potential improvements could involve adopting techniques from long-context LLMs, such as KV cache optimization, sliding memory window [59], chunk segmentation [60], and more efficient model architectures.

6. Summary

In this paper, we present a detailed methodology of ICON for PDEs with temporal evolution, using 1D scalar nonlinear conservation laws as an example. By designing the data prompts appropriately, we can use a single ICON model to make forward and reverse predictions for different equations with different strides. We show that an ICON model trained on conservation laws with cubic flux functions can generalize well to some other flux functions of more general forms, without fine-tuning. We also show how to broaden the range of problems that ICON can address, by transforming functions and equations to ICON's capability scope via change of variables and proper strides. We believe that the progress in this paper is a significant step towards the goal of training a foundation model for scientific machine learning under the in-context operator learning framework.

CRediT authorship contribution statement

Liu Yang: Writing – original draft, Visualization, Validation, Software, Resources, Project administration, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Stanley J. Osher:** Writing – review & editing, Supervision, Funding acquisition, Conceptualization.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Stanley J. Osher reports financial support was provided by Air Force Office of Scientific Research. Stanley J. Osher reports financial support was provided by Office of Naval Research. Stanley J. Osher serves as an associate editor of the Journal of Computational Physics. If there are other authors, they declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

I have shared the link to the code in the manuscript.

Declaration of generative AI and AI-assisted technologies in the writing process

During the preparation of this work the author(s) used ChatGPT in order to improve phrasing. After using this tool/service, the author(s) reviewed and edited the content as needed and take(s) full responsibility for the content of the publication.

Acknowledgement

We acknowledge valuable discussions with Prof. Chi-Wang Shu, Dr. Tingwei Meng, Dr. Siting Liu, and Yuxuan Liu. S. Osher is partially funded by AFOSR MURI FA9550-18-502 and ONR N00014-20-1-2787.

References

[1] I.E. Lagaris, A. Likas, D.I. Fotiadis, Artificial neural networks for solving ordinary and partial differential equations, IEEE Trans. Neural Netw. 9 (5) (1998) 987–1000.
[2] W. E, J. Han, A. Jentzen, Deep learning-based numerical methods for high-dimensional parabolic partial differential equations and backward stochastic differential equations, Commun. Math. Stat. 5 (4) (2017) 349–380.
[3] J. Han, A. Jentzen, W. E, Solving high-dimensional partial differential equations using deep learning, Proc. Natl. Acad. Sci. USA 115 (34) (2018) 8505–8510, https://doi.org/10.1073/pnas.1718942115.
[4] J. Sirignano, K. Spiliopoulos, DGM: a deep learning algorithm for solving partial differential equations, J. Comput. Phys. 375 (2018) 1339–1364, https://doi.org/10.1016/j.jcp.2018.08.029.
[5] W. E, B. Yu, The deep Ritz method: a deep learning-based numerical algorithm for solving variational problems, Commun. Math. Stat. 6 (1) (2018) 1–12, https://doi.org/10.1007/s40304-018-0127-z.
[6] M. Raissi, P. Perdikaris, G.E. Karniadakis, Physics-informed neural networks: a deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, J. Comput. Phys. 378 (2019) 686–707, https://doi.org/10.1016/j.jcp.2018.10.045.
[7] Y. Zang, G. Bao, X. Ye, H. Zhou, Weak adversarial networks for high-dimensional partial differential equations, J. Comput. Phys. 411 (2020) 109409, https://doi.org/10.1016/j.jcp.2020.109409, 14.
[8] L. Yang, G.E. Karniadakis, Potential flow generator with L_2 optimal transport regularity for generative models, IEEE Trans. Neural Netw. Learn. Syst. 33 (2) (2020) 528–538.
[9] L. Ruthotto, S.J. Osher, W. Li, L. Nurbekyan, S.W. Fung, A machine learning framework for solving high-dimensional mean field game and mean field control problems, Proc. Natl. Acad. Sci. USA 117 (17) (2020) 9183–9193, https://doi.org/10.1073/pnas.1922204117.
[10] A.T. Lin, S.W. Fung, W. Li, L. Nurbekyan, S.J. Osher, Alternating the population and control neural networks to solve high-dimensional stochastic mean-field games, Proc. Natl. Acad. Sci. USA 118 (31) (2021) 10, https://doi.org/10.1073/pnas.2024713118.
[11] T. Chen, H. Chen, Universal approximation to nonlinear operators by neural networks with arbitrary activation functions and its application to dynamical systems, IEEE Trans. Neural Netw. 6 (4) (1995) 911–917.
[12] T. Chen, H. Chen, Approximation capability to functions of several variables, nonlinear functionals, and operators by radial basis function neural networks, IEEE Trans. Neural Netw. 6 (4) (1995) 904–910.
[13] Y. Khoo, J. Lu, L. Ying, Solving parametric PDE problems with artificial neural networks, Eur. J. Appl. Math. 32 (3) (2021) 421–435.
[14] Y. Zhu, N. Zabaras, Bayesian deep convolutional encoder–decoder networks for surrogate modeling and uncertainty quantification, J. Comput. Phys. 366 (2018) 415–447.
[15] Z. Long, Y. Lu, X. Ma, B. Dong, PDE-net: learning PDEs from data, in: International Conference on Machine Learning, PMLR, 2018, pp. 3208–3216.
[16] Z. Li, N.B. Kovachki, K. Azizzadenesheli, B. liu, K. Bhattacharya, A. Stuart, A. Anandkumar, Fourier neural operator for parametric partial differential equations, in: International Conference on Learning Representations, 2021, https://openreview.net/forum?id=c8P9NQVtmnO.
[17] N. Kovachki, Z. Li, B. Liu, K. Azizzadenesheli, K. Bhattacharya, A. Stuart, A. Anandkumar, Neural operator: learning maps between function spaces with applications to PDEs, J. Mach. Learn. Res. 24 (89) (2023) 1–97.
[18] L. Lu, P. Jin, G. Pang, Z. Zhang, G.E. Karniadakis, Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators, Nat. Mach. Intell. 3 (3) (2021) 218–229.
[19] S. Wang, H. Wang, P. Perdikaris, Learning the solution operator of parametric partial differential equations with physics-informed DeepONets, Sci. Adv. 7 (40) (2021), eabi8605.
[20] K. Bhattacharya, B. Hosseini, N.B. Kovachki, A.M. Stuart, Model reduction and neural networks for parametric PDEs, SMAI J. Comput. Math. 7 (2021) 121–157.
[21] Z. Li, H. Zheng, N. Kovachki, D. Jin, H. Chen, B. Liu, K. Azizzadenesheli, A. Anandkumar, Physics-informed neural operator for learning partial differential equations, arXiv preprint, arXiv:2111.03794, 2021.
[22] A. Sanchez-Gonzalez, J. Godwin, T. Pfaff, R. Ying, J. Leskovec, P. Battaglia, Learning to simulate complex physics with graph networks, in: International Conference on Machine Learning, PMLR, 2020, pp. 8459–8468.
[23] T. Pfaff, M. Fortunato, A. Sanchez-Gonzalez, P. Battaglia, Learning mesh-based simulation with graph networks, in: International Conference on Learning Representations, 2020.
[24] J. Brandstetter, D.E. Worrall, M. Welling, Message passing neural pde solvers, in: International Conference on Learning Representations, 2021.
[25] D. Kochkov, J.A. Smith, A. Alieva, Q. Wang, M.P. Brenner, S. Hoyer, Machine learning–accelerated computational fluid dynamics, Proc. Natl. Acad. Sci. 118 (21) (2021) e2101784118.
[26] G. Kissas, J.H. Seidman, L.F. Guilhoto, V.M. Preciado, G.J. Pappas, P. Perdikaris, Learning operators with coupled attention, J. Mach. Learn. Res. 23 (215) (2022) 1–63.
[27] S. Goswami, K. Kontolati, M.D. Shields, G.E. Karniadakis, Deep transfer operator learning for partial differential equations under conditional shift, Nat. Mach. Intell. (2022) 1–10.
[28] M. Zhu, H. Zhang, A. Jiao, G.E. Karniadakis, L. Lu, Reliable extrapolation of deep neural operators informed by physics or sparse observations, Comput. Methods Appl. Mech. Eng. 412 (2023) 116064.
[29] A. Subel, Y. Guan, A. Chattopadhyay, P. Hassanzadeh, Explaining the physics of transfer learning in data-driven turbulence modeling, PNAS Nexus 2 (3) (2023) pgad015.

[30] S. Goswami, C. Anitescu, S. Chakraborty, T. Rabczuk, Transfer learning enhanced physics informed neural network for phase-field modeling of fracture, Theor. Appl. Fract. Mech. 106 (2020) 102447.
[31] X. Chen, C. Gong, Q. Wan, L. Deng, Y. Wan, Y. Liu, B. Chen, J. Liu, Transfer learning for deep neural network-based partial differential equations solving, Adv. Aerodyn. 3 (1) (2021) 1–14.
[32] E. Haghighat, M. Raissi, A. Moure, H. Gomez, R. Juanes, A physics-informed deep learning framework for inversion and surrogate modeling in solid mechanics, Comput. Methods Appl. Mech. Eng. 379 (2021) 113741.
[33] M. Mattheakis, H. Joy, P. Protopapas, Unsupervised reservoir computing for solving ordinary differential equations, arXiv preprint, arXiv:2108.11417, 2021.
[34] S. Chakraborty, Transfer learning based multi-fidelity physics informed deep neural network, J. Comput. Phys. 426 (2021) 109942.
[35] S. Desai, M. Mattheakis, H. Joy, P. Protopapas, S.J. Roberts, One-shot transfer learning of physics-informed neural networks, in: ICML 2022 2nd AI for Science Workshop, 2022, https://openreview.net/forum?id=1AspJ4zzeqq.
[36] Y. Gao, K.C. Cheung, M.K. Ng, SVD-PINNs: transfer learning of physics-informed neural networks via singular value decomposition, in: 2022 IEEE Symposium Series on Computational Intelligence (SSCI), IEEE, 2022, pp. 1443–1450.
[37] H. Guo, X. Zhuang, P. Chen, N. Alajlan, T. Rabczuk, Analysis of three-dimensional potential problems in non-homogeneous media with physics-informed deep collocation method using material transfer learning and sensitivity analysis, Eng. Comput. 38 (6) (2022) 5423–5444.
[38] A. Chakraborty, C. Anitescu, X. Zhuang, T. Rabczuk, Domain adaptation based transfer learning approach for solving PDEs on complex geometries, Eng. Comput. 38 (5) (2022) 4569–4588.
[39] C. Xu, B.T. Cao, Y. Yuan, G. Meschke, Transfer learning based physics-informed neural networks for solving inverse problems in engineering structures under different loading scenarios, Comput. Methods Appl. Mech. Eng. 405 (2023) 115852.
[40] H. Wang, R. Planas, A. Chandramowlishwaran, R. Bostanabad, Mosaic flows: a transferable deep learning framework for solving PDEs on unseen domains, Comput. Methods Appl. Mech. Eng. 389 (2022) 114424.
[41] W. Xu, Y. Lu, L. Wang, Transfer learning enhanced DeepONet for long-time prediction of evolution equations, in: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 37, 2023, pp. 10629–10636.
[42] Y. Lyu, X. Zhao, Z. Gong, X. Kang, W. Yao, Multi-fidelity prediction of fluid flow and temperature field based on transfer learning using Fourier Neural Operator, arXiv preprint, arXiv:2304.06972, 2023.
[43] S. Subramanian, P. Harrington, K. Keutzer, W. Bhimji, D. Morozov, M. Mahoney, A. Gholami, Towards foundation models for scientific machine learning: characterizing scaling and transfer behavior, arXiv preprint, arXiv:2306.00258, 2023.
[44] L. Yang, S. Liu, T. Meng, S.J. Osher, In-context operator learning with data prompts for differential equation problems, Proc. Natl. Acad. Sci. 120 (39) (2023) e2310142120.
[45] L. Yang, S. Liu, S.J. Osher, Fine-tune language models as multi-modal differential equation solvers, arXiv preprint, arXiv:2308.05061, 2023.
[46] Y. Liu, Z. Zhang, H. Schaeffer, Prose: predicting operators and symbolic expressions using multimodal transformers, arXiv preprint, arXiv:2309.16816, 2023.
[47] J.W. Liu, N.B. Erichson, K. Bhatia, M.W. Mahoney, C. Re, Does in-context operator learning generalize to domain-shifted settings?, in: The Symbiosis of Deep Learning and Differential Equations III, 2023.
[48] M. McCabe, B. Régaldo-Saint Blancard, L.H. Parker, R. Ohana, M. Cranmer, A. Bietti, M. Eickenberg, S. Golkar, G. Krawezik, F. Lanusse, et al., Multiple physics pretraining for physical surrogate models, in: NeurIPS 2023 AI for Science Workshop, 2023.
[49] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, et al., Language models are unsupervised multitask learners, OpenAI blog 1 (8) (2019) 9.
[50] T. Brown, B. Mann, N. Ryder, M. Subbiah, J.D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al., Language models are few-shot learners, Adv. Neural Inf. Process. Syst. 33 (2020) 1877–1901.
[51] S. Garg, D. Tsipras, P.S. Liang, G. Valiant, What can transformers learn in-context? A case study of simple function classes, Adv. Neural Inf. Process. Syst. 35 (2022) 30583–30598.
[52] Y. Bai, X. Geng, K. Mangalam, A. Bar, A. Yuille, T. Darrell, J. Malik, A.A. Efros, Sequential modeling enables scalable learning for large vision models, arXiv preprint, arXiv:2312.00785, 2023.
[53] H. Liu, C. Li, Q. Wu, Y.J. Lee, Visual instruction tuning, arXiv preprint, arXiv:2304.08485, 2023.
[54] D. Driess, F. Xia, M.S. Sajjadi, C. Lynch, A. Chowdhery, B. Ichter, A. Wahid, J. Tompson, Q. Vuong, T. Yu, et al., Palm-e: an embodied multimodal language model, arXiv preprint, arXiv:2303.03378, 2023.
[55] Q. Dong, L. Li, D. Dai, C. Zheng, Z. Wu, B. Chang, X. Sun, J. Xu, Z. Sui, A survey for in-context learning, arXiv preprint, arXiv:2301.00234, 2022.
[56] OpenAI, GPT-4 technical report, 2023, arXiv:2303.08774.
[57] X.-D. Liu, S. Osher, T. Chan, Weighted essentially non-oscillatory schemes, J. Comput. Phys. 115 (1) (1994) 200–212.
[58] J. Wei, X. Wang, D. Schuurmans, M. Bosma, F. Xia, E. Chi, Q.V. Le, D. Zhou, et al., Chain-of-thought prompting elicits reasoning in large language models, Adv. Neural Inf. Process. Syst. 35 (2022) 24824–24837.
[59] Y. Hao, Y. Sun, L. Dong, Z. Han, Y. Gu, F. Wei, Structured prompting: scaling in-context learning to 1,000 examples, arXiv preprint, arXiv:2212.06713, 2022.
[60] N. Ratner, Y. Levine, Y. Belinkov, O. Ram, I. Magar, O. Abend, E. Karpas, A. Shashua, K. Leyton-Brown, Y. Shoham, Parallel context windows for large language models, arXiv preprint, arXiv:2212.10947, 2022.