Neural Networks 188 (2025) 107455

Contents lists available at ScienceDirect

Neural Networks

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

Full Length Article

Fine-tune language models as multi-modal differential equation solvers ☆

Liu Yang, Siting Liu, Stanley J. Osher *

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

ARTICLE INFO

Keywords:
In-context learning
Multi-modal machine learning
Operator learning
Differential equation

ABSTRACT

In the growing domain of scientific machine learning, in-context operator learning has shown notable potential in building foundation models, as in this framework the model is trained to learn operators and solve differential equations using prompted data, during the inference stage without weight updates. However, the current model's overdependence on function data overlooks the invaluable human insight into the operator. To address this, we present a transformation of in-context operator learning into a multi-modal paradigm. In particular, we take inspiration from the recent success of large language models, and propose using "captions" to integrate human knowledge about the operator, expressed through natural language descriptions and equations. Also, we introduce a novel approach to train a language-model-like architecture, or directly fine-tune existing language models, for in-context operator learning. We beat the baseline on single-modal learning tasks, and also demonstrated the effectiveness of multi-modal learning in enhancing performance and reducing function data requirements. The proposed method not only significantly enhanced the development of the in-context operator learning paradigm, but also created a new path for the application of language models.

1. Introduction

Recently, in-context operator learning and the corresponding model In-Context Operator Networks (ICON) (Yang et al., 2023) has been proposed as a new paradigm for scientific machine learning.

As in classic operator learning tasks, an operator maps a single input function or a tuple of input functions, referred to as the "condition", to an output function, referred to as the "quantity of interest (QoI)". In practice, we usually have no access to the analytical expression of these functions, but instead can collect function data in the form of key–value pairs, where the keys are discrete function inputs and the values are the corresponding function outputs.

A wide variety of scientific machine learning tasks can be conceptualized as operator learning problems. Consider the task of solving partial differential equations (PDEs) for instance, where the coefficient function serves as the condition, and the solution is the QoI. Conversely, for inverse problems, these roles are swapped. When dealing with problems involving temporal evolution, the condition can be the initial function, while the QoI represents the function at a later time. For control problems, the condition could correspond to the cost function and the initial state, while the QoI embodies the control signal. It is evident that the relationship between the condition and the QoI highly depends on the operator, which is defined by the task at hand and the particular system in question.

In classic operator learning approaches (Bhattacharya et al., 2021; Chen & Chen, 1995a, 1995b; Khoo et al., 2021; Kovachki et al., 2023; Li, Kovachki, et al., 2021; Li, Zheng, et al., 2021; Long et al., 2018; Lu et al., 2021; Subramanian et al., 2023; Wang et al., 2021; Zhu & Zabaras, 2018), a neural network is limited to approximate a specific operator, and thus need to be trained every time a new operator is encountered. In contrast, in-context operator learning aims to train the model as an "operator learner" instead of an "operator approximator". In particular, the model is trained to learn the operator from the prompted condition-QoI pairs, referred to as "examples", and apply the learned operator to the question condition to predict the corresponding QoI. After training, the above learning process can be performed through one forward pass of the model, in the inference stage without weight update. This approach offers a "train-once-apply-multiple" paradigm and paves the way for large-scale foundation models (Bommasani et al., 2021) for a broad array of scientific machine learning tasks.

The study of ICON showcases the successful implementation of in-context operator learning, which relies solely on numerical function data. However, a crucial aspect of scientific machine learning is overlooked in this approach, namely, the human knowledge of the operator, which can span from vague natural language explanations to explicit

☆ All code is deposited in https://github.com/LiuYangMage/in-context-operator-networks.

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

https://doi.org/10.1016/j.neunet.2025.107455
Received 17 May 2024; Received in revised form 27 December 2024; Accepted 30 March 2025
Available online 21 April 2025

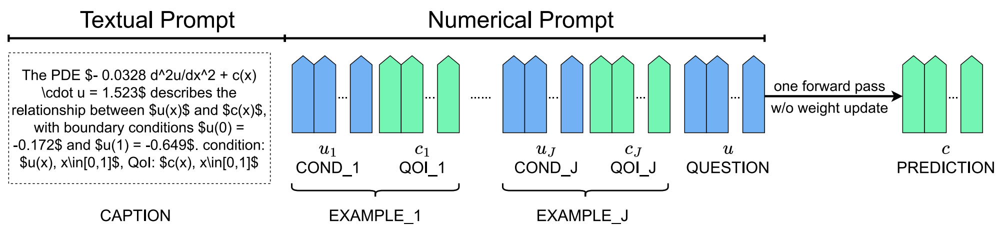


Fig. 1. Diagram for multi-modal in-context operator learning.

differential equations. There is a strong case for incorporating such knowledge into the learning system alongside numerical data, as this could potentially enhance learning performance with a fixed budget of numerical data. Moreover, sometimes we may not have any numerical examples available. For these cases, i.e., zero-shot learning, human knowledge would be necessary for the model to identify the problem.

Past research on the topic of scientific machine learning typically integrates human knowledge into the learning system by designing special loss functions or neural network architectures based on the differential equations or symmetry/conservation laws that govern the system. While these approaches have witnessed significant success, they are not without limitations. Firstly, it may not always be practical to design special loss functions or architectures, as the system might not be fully understood by humans, or the operator might be too complicated to be described by equations. Secondly, these bespoke loss functions or architectures are tailored for specific systems or tasks. When confronted with a new system, there is a requirement not only to design new loss functions or architectures but also typically to retrain the neural network. Thirdly, these loss functions or architectures integrate human knowledge in the training stage, and is not straightforward to be applied in the inference stage, thus limiting the application of these methods in the "train-once-apply-multiple" paradigm.

In this paper, we explore an entirely different approach to infusing human knowledge into the learning system. Inspired by the recent success of large language models (LLMs), we introduce a new component to in-context operator learning: the "caption". A caption is a string serving as a descriptor of the operator, and can take various forms such as equations written in LaTeX forms, natural language descriptions, or a combination of both. Rather than crafting special loss functions or architectures, we simply feed the caption into the neural network as input alongside the examples. We thus evolve the in-context operator learning to be multi-modal, meaning that the neural network can learn the operator from numerical data, captions, or a combination of both, as illustrated in Fig. 1. Multi-modal in-context operator learning improves the accuracy given a fixed number of numerical examples, and more importantly, enables zero-shot learning with no numerical examples. Moreover, captions overcome the limitations of the aforementioned methods of integrating human knowledge, in that (1) the integrated human knowledge can range from vague to precise, (2) the method is general and flexible to be applied to various systems or tasks, and (3) the method is applicable in the inference stage, aligning seamlessly with the "train-once-apply-multiple" paradigm.

We also introduce a novel approach to train a language-model-like architecture, or directly fine-tune existing language models, for single-modal and multi-modal in-context operator learning. The improved training scheme mimics the "next-token prediction" in LLMs: the model predicts the QoI in each example based on previous examples. We call it "next-function prediction". The main deviation (and also the key challenge) is the necessity to design the input sequence and formulate a specialized mask to accommodate in-context operator learning tasks. Following the name "In-Context Operator Networks (ICON)", we refer to our architecture and training scheme as "ICON-LM", where "LM" stands for "language model".

The adoption of language models for in-context learning is crucial for two reasons. First, it enables us to utilize existing ecosystem developed for language models for in-context operator learning. Second, it paves the way to broaden the capability of language models to scientific machine learning tasks with heavy numerical computations.

Our contributions are summarized as follows:

1. We transform the in-context operator learning into a multi-modal framework by introducing "captions" as a means to incorporate human knowledge about the operator, in the form of natural language descriptions and equations.
2. We introduce a novel approach, namely "ICON-LM", to train a language-model-like architecture, or directly fine-tune existing language models, for in-context operator learning. We outperformed the baseline on single-modal learning tasks, and also demonstrated the effectiveness of multi-modality with ICON-LM in enhancing performance and reducing numerical data requirements.
3. By bridging language models with data-driven differential equation solvers, we have not only achieved substantial advancements in this specific domain of operator learning, but also opened up a new avenue for the application of language models in scientific machine learning tasks that require heavy numerical computations.

The rest of the paper is organized as follows. In Section 2, we review the related work. We introduce the dataset in Section 3. In Section 4, we introduce the ICON-LM architecture and training scheme. In Section 5, we present the experimental results. We conclude in Section 6.

2. Related work

2.1. Operator learning and in-context operator learning

Numerous neural network methods have been proposed for approximating operators, i.e., mappings that take functions as input and output. The early works of Chen and Chen (1995a, 1995b) employed shallow neural networks for the approximation of nonlinear operators. A deep neural network approach to tackle parametric PDE challenges was suggested in Khoo et al. (2021). PDE-Net, as presented in Long et al. (2018) enables forward predictions of PDE solutions using the inferred forward map. The study in Zhu and Zabaras (2018) presented a Bayesian method to address uncertainty quantification in stochastic PDE scenarios. The Deep Operator Network (DeepONet), referenced in Lu et al. (2021), introduces a neural network design that approximates the solution operator, mapping parameters or initial/boundary conditions to their corresponding solutions. The Fourier Neural Operator (FNO) from Kovachki et al. (2023), Li, Kovachki, et al. (2021) leverages the Fourier space's integral kernel to approximate the solution operator. Drawing inspiration from neural networks and model reduction, Bhattacharya et al. (2021) estimates input–output maps between infinite-dimensional spaces for parametric PDEs. Additional contributions can be found in Goswami et al. (2022), Kissas et al. (2022), Kochkov et al. (2021), Subel et al. (2023), Zhu, Zhang, et al. (2023).

Recently, a different paradigm, namely in-context operator learning, is proposed in Yang et al. (2023), which is an extension of in-context learning introduced in GPT-2 (Radford et al., 2019) and GPT-3 (Brown et al., 2020). Instead of approximating specific operators, in-context operator learning trains the neural network as an operator learner, which can learn and apply the operator through one forward pass of the model, in the inference stage without weight update. Such in-context learning capability can even generalize to new equations (Liu, Erichson, et al., 2023; Yang & Osher, 2024).

2.2. Physics-informed machine learning

In the literature, two approaches are commonly employed to incorporate physical knowledge in neural networks: hard constraints and soft constraints. We refer readers to the survey paper (Karniadakis et al., 2021) on this topic. Hard constraints involve designing neural network architectures in a way that ensures any solution generated by the network meets the specified constraints, for example, Jin et al. (2020), Ling et al. (2016), Lusch et al. (2018), Mattheakis et al. (2019), Pfau et al. (2020), Pun et al. (2019), Zhang et al. (2018). While solutions with specifically designed architectures are guaranteed to be compliant to the physical constraints, creating such architectures demands extensive domain knowledge and may not be easily adaptable to other problems. Additionally, the expressivity and training complexity could be limited in these cases. Soft constraints are implemented by incorporating physics-informed terms into the loss function. For example, E et al. (2017), E and Yu (2018), Han et al. (2018), Li, Zheng, et al. (2021), Raissi et al. (2019), Ruthotto et al. (2020), Sirignano and Spiliopoulos (2018), Wang et al. (2021), Zang et al. (2020). While more flexible in terms of neural network architecture design, this approach still requires precise knowledge of physics in the form of differential equations, variational problems, etc., which are not always available, especially when the system is not fully understood by humans.

In-context operator learning excels at addressing a broad range of physical problems using a single neural network. The limited flexibility and generalizability of the previously mentioned approaches hinder their application to in-context operator learning. This limitation motivates our exploration in this paper, where we introduce a new method to incorporate physical knowledge: through "captions".

2.3. Multi-modal models

Unimodal language models solely rely on text data for training, limiting their ability to comprehend the visual world. In contrast, multimodal language models are trained on data in multiple forms, including texts and images, enabling them to understand the visual world. We refer readers to the survey Yin et al. (2023) on this topic.

To fuse different modal data, one approach involves combining the extracted features or embeddings from different modal data and then feeding these embeddings into the same model (Alayrac et al., 2022; Brohan et al., 2023; Driess et al., 2023; Li, Li, et al., 2023; Liu, Li, et al., 2023; Pi et al., 2023; Tsimpoukelli et al., 2021; Zhang, Han, et al., 2023; Zhang, Li, & Bing, 2023; Zhang, Wu, et al., 2023; Zhu, Chen, et al., 2023). Another approach converts other modal data into language data and uses these language representations as inputs for language models (Yang et al., 2022). Some studies combine both techniques, utilizing both extracted features and converted language data as inputs to language models (Gao et al., 2023; Li, He, et al., 2023).

In the domain of scientific machine learning, multi-modal learning is also applied to merge numerical and symbolic data (Liu, Zhang, & Schaeffer, 2023; Ye et al., 2024).

3. Dataset

The dataset in this research work consists of two modes: numerical data and textual captions. For single-modal learning, only numerical data are used, while multi-modal learning involves both modes.

In this study, we use the numerical dataset from Yang et al. (2023). This dataset contains 19 types of operator learning problems, including forward and inverse ordinary differential equations (ODEs), partial differential equations (PDEs), and mean-field control (MFC) problems. Notably, each type is characterized by a set of hidden parameters that define the operator, meaning that each type comprises an infinite number of operators. We list the 19 types of problems in Table 1.

Within the training dataset, each problem type comes with 1000 distinct operators characterized by hidden parameters. For every operator, there are 100 condition-QoI pairs governed by such a shared operator. During training, one can randomly sample from these pairs as "examples" to build an instance of prompt and label.

In the testing dataset, each problem type is represented by an additional 100 unique operators. Every operator is associated with 5 sets of condition-QoI pairs, and each set has 6 such pairs. For testing purposes, the initial J pairs in each set can serve as "examples", while the final pair acts as the "question" for J-shot learning, with J ranging from zero to five. This means the testing dataset consists of $19 \times 100 \times 5$ sets, translating to $19 \times 100 \times 5$ learning cases for every value of J.

In these problems, the condition/QoI function is in a 1D or 2D domain, depending on the problem type. We can sample data points from discrete grids for each function to construct data prompts that represent the function.

For multi-modal learning, per problem type, we produced 160 captions for training and an additional 40 for testing. These captions are evenly split into two categories: vague and precise, depending on whether they reveal the actual parameter values that determine the operator, e.g., the decay rate of a damped oscillator, the boundary condition of PDEs, or the terminal cost in a mean-field control problem.

These caption data are generated with the assistance of GPT-4. In short, the GPT-4 model is prompted to rephrase a few handcrafted captions following instructions, with parameter placeholders in precise captions. The technical details of caption data generation in Appendix A. All the captions are open-sourced alongside code, with some examples listed in Appendix B.

An illustration of training and testing datasets is provided in Fig. 2.

4. ICON-LM model

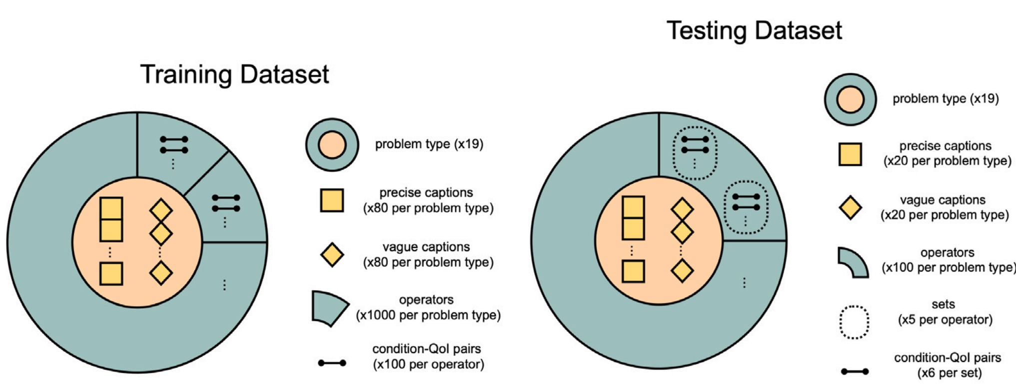
4.1. Overview

The original ICON architecture consists of two transformers: an encoder and a decoder. For every prompt instance, the model is solely trained to prediction one QoI function, with a given number of examples. Specifically, the encoder processes these examples along with an additional "question condition" to produce a sequence of embeddings. Following this, the decoder ingests these embedding and "queries", which represent the spatial–temporal coordinates, to predict the QoI function associated to the question condition and evaluated at these query points.

We identified this architecture and training method as inefficient. In this paper, we propose to replace the encoder–decoder architecture with a decoder-only transformer architecture, and perform in-context operator learning in an autoregressive manner. In particular, the model predicts each QoI function in the prompt conditioned on previous examples, and optionally the caption. We term this training scheme as "next-function prediction", in parallel to the"next-token prediction" training scheme in language models. "next-function prediction" is more efficient than the original ICON, as for each prompt instance, the

Table 1
List of differential equation problems studied in this work.

#	Problem type	Differential equations	Parameters	Conditions	QoIs
1	Forward ODE 1	$\frac{d}{dt}u(t) = a_1 c(t) + a_2$ for $t \in [0,1]$	a_1, a_2	$u(0), c(t), t \in [0,1]$	$u(t), t \in [0,1]$
2	Inverse ODE 1			$u(t), t \in [0,1]$	$c(t), t \in [0,1]$
3	Forward ODE 2	$\frac{d}{dt}u(t) = a_1 c(t)u(t) + a_2$ for $t \in [0,1]$	a_1, a_2	$u(0), c(t), t \in [0,1]$	$u(t), t \in [0,1]$
4	Inverse ODE 2			$u(t), t \in [0,1]$	$c(t), t \in [0,1]$
5	Forward ODE 3	$\frac{d}{dt}u(t) = a_1 u(t) + a_2 c(t) + a_3$ for $t \in [0,1]$	a_1, a_2, a_3	$u(0), c(t), t \in [0,1]$	$u(t), t \in [0,1]$
6	Inverse ODE 3			$u(t), t \in [0,1]$	$c(t), t \in [0,1]$
7	Forward damped oscillator	$u(t) = A\sin(\frac{2\pi}{T}t + \eta)e^{-kt}$ for $t \in [0,1]$	k	$u(t), t \in [0,0.5)$	$u(t), t \in [0.5,1]$
8	Inverse damped oscillator			$u(t), t \in [0.5,1]$	$u(t), t \in [0,0.5)$
9	Forward Poisson equation	$\frac{d^2}{dx^2}u(x) = c(x)$ for $x \in [0,1]$	$u(0), u(1)$	$c(x), x \in [0,1]$	$u(x), x \in [0,1]$
10	Inverse Poisson equation			$u(x), x \in [0,1]$	$c(x), x \in [0,1]$
11	Forward linear reaction–diffusion	$-\lambda a \frac{d^2}{dx^2}u(x) + k(x)u(x) = c$ for $x \in [0,1]$, $\lambda = 0.05$	$u(0), u(1), a, c$	$k(x), x \in [0,1]$	$u(x), x \in [0,1]$
12	Inverse linear reaction–diffusion			$u(x), x \in [0,1]$	$k(x), x \in [0,1]$
13	Forward nonlinear reaction–diffusion	$-\lambda a \frac{d^2}{dx^2}u(x) + ku(x)^3 = c(x)$ for $x \in [0,1]$, $\lambda = 0.1$	$u(0), u(1), k, a$	$c(x), x \in [0,1]$	$u(x), x \in [0,1]$
14	Inverse nonlinear reaction–diffusion			$u(x), x \in [0,1]$	$c(x), x \in [0,1]$
15	MFC g-parameter 1D → 1D	$\inf_{\rho,m} \iint c\frac{m^2}{2\rho}dxdt + \int g(x)\rho(1,x)dx$ s.t. $\partial_t \rho(t,x) + \nabla_x \cdot m(t,x) = \mu \Delta_x \rho(t,x)$ for $t \in [0,1], x \in [0,1]$, $c = 20, \mu = 0.02$, periodic spatial boundary condition	$g(x), x \in [0,1]$	$\rho(0,x),\ x \in [0,1]$	$\rho(1,x), x \in [0,1]$
16	MFC g-parameter 1D → 2D			$\rho(0,x),\ x \in [0,1]$	$\rho(t,x),\ t \in [0.5,1],\ x \in [0,1]$
17	MFC g-parameter 2D → 2D			$\rho(t,x), t \in [0,0.5), x \in [0,1]$	$\rho(t,x), t \in [0.5,1], x \in [0,1]$
18	MFC ρ_0-parameter 1D → 1D		$\rho(0,x)$, $x \in [0,1]$	$g(x), x \in [0,1]$	$\rho(1,x),\ x \in [0,1]$
19	MFC ρ_0-parameter 1D → 2D				$\rho(t,x)$,$t \in [0.5,1]$,$x \in [0,1]$



(a) (b)

Fig. 2. Illustration of (a) training dataset and (b) testing dataset.

model simultaneously performs in-context operator learning with varying numbers of examples, ranging from zero (with a caption) or one (without a caption) up to the maximum capacity. This is written as

$$\{\text{prediction of } \mathrm{QOI}_i\}_{i=1}^{I} = \mathcal{T}_\theta[\{\mathrm{CAPT}, \langle \mathrm{COND}_i, \mathrm{QOI}_i\rangle\}_{i=1}^{I}],$$
$$\text{prediction of } \mathrm{QOI}_{J+1} = \mathcal{T}_\theta[\mathrm{CAPT}, \mathrm{COND}_{J+1}, \{\langle \mathrm{COND}_i, \mathrm{QOI}_i\rangle\}_{i=1}^{J}], \qquad (1)$$
$$J = 0, 2, \ldots, I-1.$$

for in-context operator learning with captions, and

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

for in-context operator learning without captions, where $\mathcal{T}_\theta$ is the ICON-LM model, CAPT is the caption, COND_i and QOI_i are the condition and

Table 2
The tokens for the jth example for the one-dimensional forward ODE problem.

		condition					QoI				query			
key	term	0	0	…	0	1	0	0	…	0	0	0	…	0
	time	t_1	t_2	…	t_{n_j-1}	0	τ_1	τ_2	…	τ_{m_j}	τ_1	τ_2	…	τ_{m_j}
	space	0	0	…	0	0	0	0	…	0	0	0	…	0
value		$c(t_1)$	$c(t_2)$	…	$c(t_{n_j-1})$	$u(0)$	$u(\tau_1)$	$u(\tau_2)$	…	$u(\tau_{m_j})$	0	0	…	0

QoI in ith example, respectively. For $J = 0$, $\{\langle \text{COND}_i, \text{QOI}_i \rangle\}_{i=1}^{J}$ means no examples.

While drawing parallels between language models and in context operator learning, it is essential to underscore the distinctions and unique requirements of in-context operator learning: (1) The model should be invariant to the permutation of tokens within a function, since these tokens are unordered. (2) The prediction of a QoI function should not be limited to a preset collection of function inputs but is applicable to any inputs. (3) For a QoI function, the outputs corresponding to specific queries should not be generated sequentially as in language models. Rather, these predictions should be made in parallel and independent of each other.

To address these challenges, we need to design customized input and output sequences, as well as a transformer mask, which will be discussed in the following subsections.

4.2. Input tokens

The input sequence, or "prompt", of ICON-LM model consists of two parts: textual captions and numerical function data. The textual captions are tokenized in the same ways as in language models. As for the numerical part, each condition or QoI function is represented by a sequence of numerical tokens, each representing a data point for the function. In particular, each numerical token is the concatenation of a key–value pair, where conceptually the key is the function input, including temporal and spatial coordinates, and the value is the corresponding function outputs. Sometimes a condition/QoI consists of multiple terms. For example, in forward ODE problems, the condition consists of the control function as well as the initial condition. To handle such scenarios, we also include a "term" indicator to the key to distinguish multiple terms.

In the original ICON, apart from the key–value pair, a one-hot index vector is also included in each numerical token, but no positional encodings is applied. Actually, "concatenating the one-hot index vector in tokens" is equivalent to "adding positional encodings suitable" (as introduced later). We thus drop the index vector and only use the key and value in function tokens for simplicity.

The model predicts each QoI function based on the previous examples. Crucially, these predictions should be made for any function inputs, in parallel and independent of each other. To address such requirements, in addition to the condition and QoI function tokens, we also include the "query tokens" in the sequence, which are the vectors representing the keys of the QoI function. Unlike the approach in the encoder–decoder ICON, where queries are created solely for the last example, in our method, queries are created for each example.

As a demonstration, in Table 2, we show the tokens of the jth example for the one-dimensional forward ODE problem, where the condition consists of the control $c : [0, T] \to \mathbb{R}$ and the initial condition $u(0)$; the QoI is the state $u : [0, T] \to \mathbb{R}$. In the table, each column represents a token. We use $n_j - 1$ key–value pairs to represent c, one key–value pair for $u(0)$, and m_j key–value pairs for u. In the first row, we use the indicators 0 and 1 to distinguish different terms in the condition, i.e., c and $u(0)$. The third row is populated with zeros since there are no spatial coordinates in this problem. During training, the keys in query tokens are the same as those for QoIs, but the values are populated with zero.

4.3. Model and attention mask

As in language models, every caption token is transformed into an embedding vector via an embedding layer. Similarly, every condition, QoI, and query token is transformed into an embedding vector via a linear embedding layer shared by these numerical tokens. All these embedding vectors are concatenated to a sequence.

Before being supplied to the transformer, the sequence is added with positional encodings. For the caption part, we apply positional encodings as in the GPT-2 model. We also add learnable positional encodings for the function part. Notably, we not only need to distinguish different examples and different types of tokens (whether condition, QoI, or query), but also ensure that the model remains invariant to the order of tokens within a function. Therefore, all the tokens within the same condition/QoI/query share the same positional encoding. For example, with five condition-QoI pairs ($I = 5$ in Eq. (1) and Eq. (2)), there would be a total of 15 learnable positional encoding vectors designated for functions: five each for condition tokens, QoI tokens, and query tokens.

After being added with positional encodings, the input sequence is supplied to a transformer. The output sequence of the transformer is then fed into a head layer (e.g., a linear layer in this paper), to align the dimensions with those of the QoI values.

In the output sequence, we only keep the ones corresponding to the query tokens, since these parts aim to predict the QoI function values evaluated at the queries. For example, if the QoI represents function u, the output corresponding to the query token x aims to predict $u(x)$.

The input/output sequence and the model architecture are depicted in Fig. 3.

The design of the transformer mask is the key challenge in the ICON-LM model due to the following constraints. (1) The model predicts the QoI value, taking into account all the caption tokens, all the conditions and QoI tokens from previous examples, all the condition tokens of the current example, as well as the current query token. (2) When making the prediction, it is crucial to prevent inadvertent leakage of the QoI tokens in the current example, which contain the prediction targets. (3) Also, the queries should not attend to each other, as the predictions should be independent. (4) The invariance to the permutation of tokens within a function should be maintained. (5) In the end, we need to ensure that there is no unintentional information leakage due to indirect attention. This can be verified by $bool(MM) = M$ where M is the attention matrix and $bool(\cdot)$ converts each entry of the matrix to bool.

We carefully designed the mask that satisfies all the constraints above, illustrated in Fig. 4. The mask block for caption tokens is lower triangular, in consistency with the existing generative language model. The other blocks are not lower triangular for the sake of permutation invariance. The blocks for queries are diagonal, indicating that the query tokens do not attend to each other.

Since the model architecture is similar to that of language models, we can directly fine-tune existing language models for in-context operator learning, especially for multi-modal learning. The only changes are the embedding layer for function tokens and the head layer, as well as the customized transformer mask.

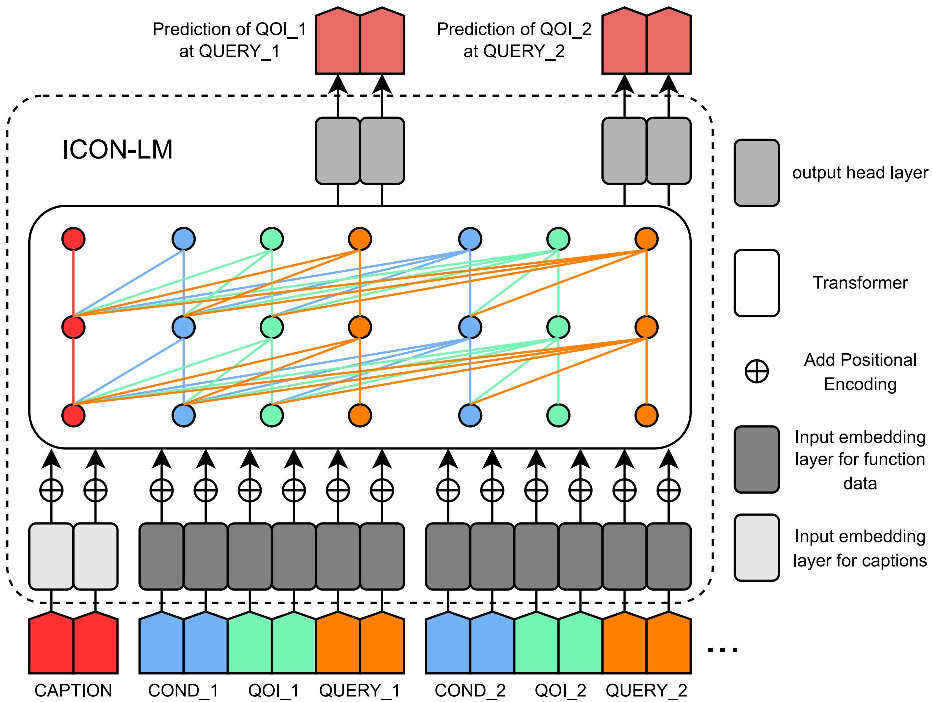


Fig. 3. Depiction of the input/output sequence and model architecture of ICON-LM. The connections in the transformer block are a simplified illustration of the attention mask.

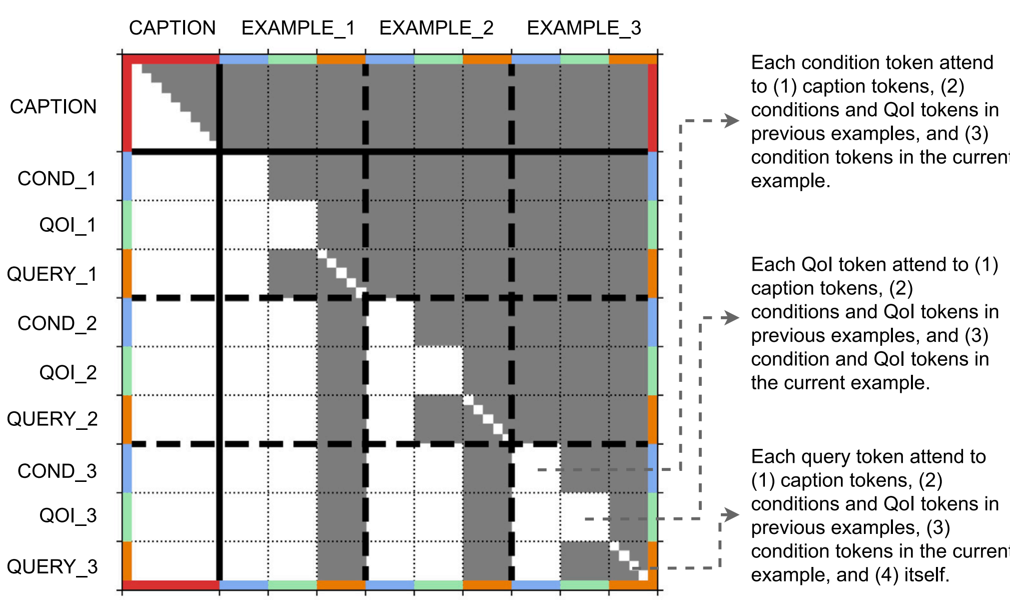


Fig. 4. The transformer mask for ICON-LM with three condition-QoI pairs. White cells representing ones, and gray cells representing zeros.

4.4. Training and inference

We can train the ICON-LM model to execute in-context operator learning, with the option of including or excluding captions. The loss function is the mean squared error between the predicted QoI values and the ground truth. For training inclusive of captions, the loss function is calculated from the first example prediction up to the last, with the first example prediction being a zero-shot – a prediction solely based on the caption and condition, excluding any other examples. When training without captions, we exclude the caption from the input sequence and calculate the loss function from the second example's predictions to the last, bypassing zero-shot learning as predicting the QoI value without any example or caption is not meaningful. The total loss for multi-modal training comprises the losses from both options.

During inference, provided with a few example condition-QoI pairs and a question condition, we want to predict the QoI corresponding to the question condition. Importantly, the prediction of the QoI should be feasible at any location within the domain, rather than being limited to predetermined fixed positions. ICON-LM efficiently achieves this by constructing question query tokens, where the keys represent where we aim to evaluate the predicted QoI. Owing to a carefully designed mask, these query tokens operate independently, and a flexible number of them is allowed. The question condition and query tokens are then appended to the input sequence as the “last example”. The QoI

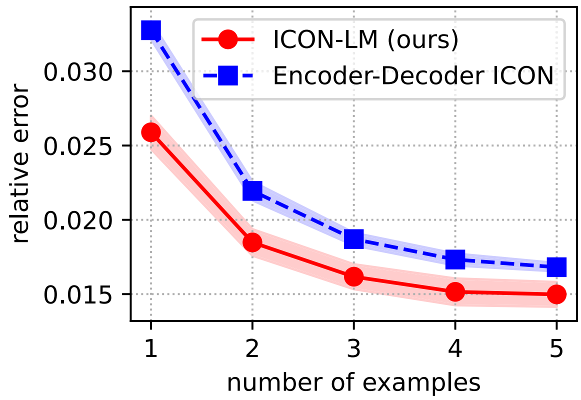


Fig. 5. Comparison of ICON-LM (ours) and encoder–decoder ICON for single-modal in-context operator learning. We calculate the relative testing error averaged over all 19 types of problems, and take the mean and standard deviation over three runs, shown as the solid line and the shaded area, respectively.

tokens for the question are not required, so are all the query tokens in examples since we do not need to predict example QoIs during inference.

5. Experiments

5.1. ICON-LM v.s. Encoder-decoder ICON

This section serves to show the performance of ICON-LM for single-modal in-context operator learning, i.e., without captions, and compare it with the baseline, the encoder–decoder ICON model.

The encoder–decoder ICON is trained with J examples alongside one question, with J randomly selected between one and five for each prompt instance. In contrast, ICON-LM is trained with six examples per instance, allowing for concurrent one-shot to five-shot learning. During both training and testing, each condition or QoI has 41 to 50 tokens. We emphasize that although the number of examples varies between the two models, the total training dataset remains consistent for both, which are presented in Section 3.

Both models are trained with the same setups for optimizer and learning rate schedule. More details on the model sizes and training configurations are given in Appendix C. The encoder–decoder ICON encompasses approximately 31.6 million parameters, whereas the ICON-LM has nearly half that number, at around 15.8 million. This substantial reduction is credited to the ICON-LM's simplified architecture, which employs a single transformer encoder roughly equivalent in size to the encoder or decoder in the baseline ICON. Compared with the baseline ICON, the larger sequence length (about $\times 1.5$) in ICON-LM requires more GPU memory, but this is largely offset by the single transformer encoder design, and slightly smaller batch size (32 for encoder–decoder ICON, and 24 for ICON-LM). With such setups, both models take about 19 GB GPU memory, and can fit in one NVIDIA GeForce RTX 4090 GPU with 24 GB memory. As for the time consumption, the training takes about 41.5 h for the encoder–decoder ICON, and about 37.5 h for ICON-LM.

We compare the relative testing error from one-shot to five-shot learning. Here the relative error is defined in the same way as in Yang et al. (2023): the absolute error is the mean difference between the predicted QoI values and the ground truth, the relative error is the absolute error divided by the mean of the absolute values of the ground truth. The comparison results are shown in Fig. 5. It is clear that ICON-LM consistently outperforms baseline encoder–decoder ICON.

As an illustration, an instance of Problem #17 is illustrated in Fig. 6. In this case, we need to infer the operator defined by the hidden terminal cost $g(x)$ from merely three examples (without captions), and then apply it to the question condition, the density field during $t \in [0, 0.5]$, to make predictions of the density field during a later time interval. Note that in the prompt, there are only 41 to 50 scattered data points for each condition/QoI function, but we can make the prediction in the whole domain of $(t, x) \in [0.5, 1] \times [0, 1]$.

5.2. In-context operator learning v.s. Classic operator learning

How does in-context operator learning compare with classic operator learning, especially when the data is limited? We examine ICON-LM against classic operator learning methods, including FNO (Li, Kovachki, et al., 2021) and DeepONet (Lu et al., 2021). Specifically, we use the example of Problem #14 out of 19 types, namely the inverse nonlinear reaction–diffusion PDE problem, and focus on the five-shot learning scenario. Each condition $u(x)$ and QoI $c(x)$ is represented by 101 evenly spaced data points in $x \in [0, 1]$.

The ICON-LM model is inherited from Section 5.1. While trained with 41 to 50 tokens for each condition/QoI, it can generalized to 101 tokens without any fine-tuning. Such generalization capability can be attributed to the flexible input sequence length of transformers, and is also reported in Yang et al. (2023).

To enable few-shot learning, we pretrain the FNO and DeepONet models using the training dataset designated for Problem #14, which comprises a distribution of operators denoted as $\mathcal{P}$. In the pretraining, these models aim to approximate the mean operator and predict $\mathbb{E}_{T\sim\mathcal{P}}T(u)$ for a given condition u. Then for each testing operator $T^* \sim \mathcal{P}$, we fine-tune the pretrained models using five examples to approximate T^*. The details of operator learning models, pretraining and fine-tuning configurations are in Appendix C. We note that the parameter numbers of FNO and DeepONet are comparable to or slightly larger than ICON-LM. The comparison is illustrated in Fig. 7.

The pretrained operator learning models effectively approximate the mean operator, and as a result, the fine-tuned models show a satisfactory approximation of the testing operator T^* when T^* is close to the mean operator, as shown in Fig. 7(b). However, their performance deteriorates when attempting to approximate T^* that deviates from the mean, as shown in Fig. 7(c). In contrast, ICON-LM consistently performs well due to its in-context learning capability for a distribution of operators. This is highlighted when considering the relative error averaged over 500 testing cases, as depicted in Fig. 7(a). The error of ICON-LM with just one forward pass for each testing case is substantially lower compared with FNO and DeepONet fine-tuned for each testing case.

5.3. Multi-modal in-context operator learning

In this section, we demonstrate the effectiveness of multi-modal in-context operator learning. We full-parameter fine-tuned the GPT-2 model with 124M parameters (Radford et al., 2019) in the ICON-LM framework. The training setup is the same as ICON-LM in Section 5.1, except that the total loss for multi-modal training comprises the losses with captions and without captions, and the batch size is reduced to 10. The training takes about 5.5 days on dual NVIDIA GeForce RTX 4090 GPUs.

In Fig. 8, we show the relative testing errors averaged over all 19 types of problems from one-shot to five-shot learning, examining the performance when prompted without captions, with vague captions, and with precise captions. A comprehensive comparison for each specific problem type can be found in Fig. 9.

In Fig. 8 we can see that precise captions improve the performance with a fixed number of numerical examples. In particular, precise captions significantly improves the zero-shot learning performance. This is because for most of the problems, there is no way to identify the operator just from a question condition, but the precise captions can switch the operators to be identifiable. Indeed, we can see from Fig. 9 that for most of the problems, zero-shot learning with precise captions achieves accuracy comparable to using five examples. This suggests

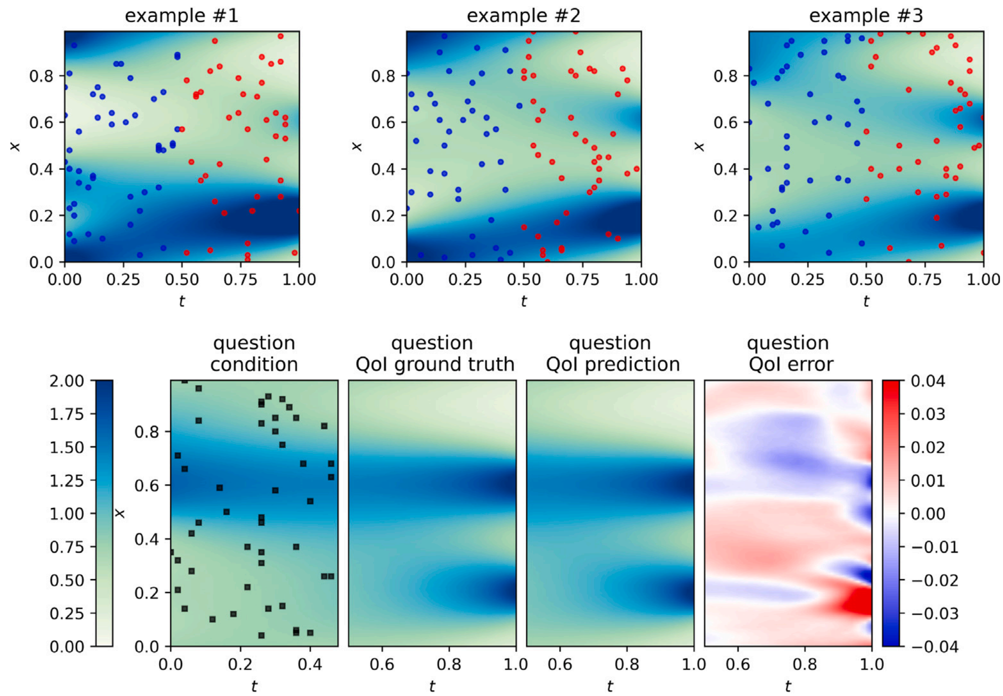


Fig. 6. Illustration of an in-context learning case in Problem #17. The blue/red round dots represent the data points for example conditions/QoIs in the prompt; the black square dots represent the data points for the question condition in the prompt.

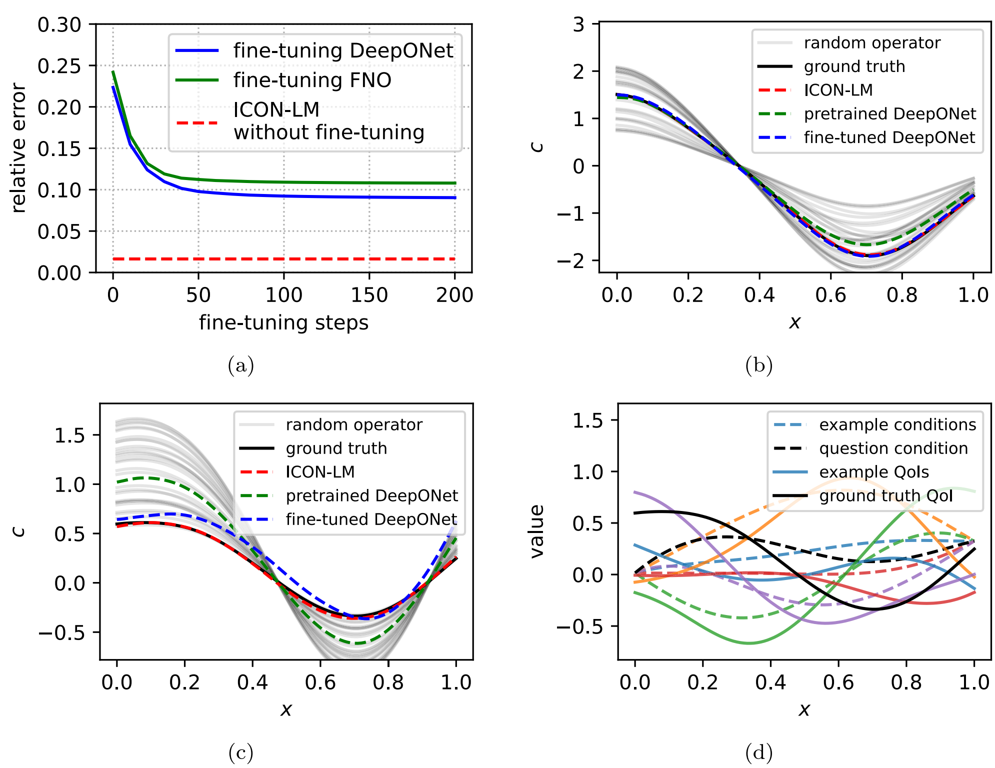


Fig. 7. Comparison of ICON-LM against FNO and DeepONet. (a) relative error during fine-tuning FNO and DeepNet. (b) prediction for a testing operator close to the mean operator. (c) prediction for a testing operator far from the mean operator. (d) five examples and the question condition for the testing operator in (c).

that the model can infer nearly the full information of the operator solely from the captions. For mean-field problems, zero-shot learning with precise captions is less effective. In these cases, the parameters are functions represented by 10 discrete points in the captions, which are insufficient to capture the full information of the operator.

The results with vague captions are also very interesting. If we take a close look at Fig. 9, we can see that vague captions also significantly

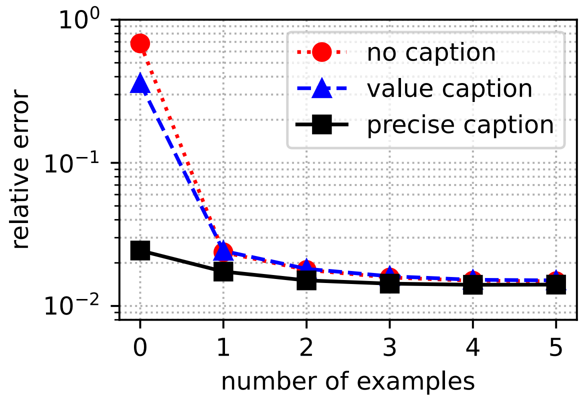


Fig. 8. Comparison of multi-modal ICON-LM with vague captions, precise captions, and no captions.

Table 3
Zero-shot relative errors of ICON-LM with pretrained and unpretrained GPT-2 model.

Model	Train (%)	Test (%)	Gap (%)
Pretrained	2.30	**2.44**	**0.14**
Unpretrained	**2.07**	3.08	1.01

improve the performance of zero shot learning for some problems. e.g. the forward and inverse damped oscillator series problems, and the inverse Poisson equation problem. This is because for these problems, once the problem type is identified with the vague captions, the operator parameters can be inferred from the conditions provided. In other words, the operators become identifiable with the *combination* of vague captions and numerical conditions. For instance, the decay rate of the damped oscillator can be computed from a segment of the time series in the given condition, and the boundary conditions of the Poisson equation are encapsulated within the conditions of $u(x)$. These examples highlight the impressive capabilities of the ICON-LM framework in combining multi-modal information to learn the operator.

5.4. Ablation study

Here we conduct an ablation study to inspect the differences between pretrained and unpretrained language models within the multi-modal in-context operator learning framework. The training setup is the same as in Section 5.3, with the sole variable being the initial state of the GPT-2 model: either pretrained or initialized randomly.

In Table 3 we show the results for zero-shot learning with precise captions, where the relative error is averaged across all 19 problem types. The results reveal that while the unpretrained GPT-2 model performs better on the training captions, it generalizes poorly to the testing captions. Meanwhile, the generalization gap is much smaller for the pretrained GPT-2 model. This shows that a pretrained language model enhances the ICON-LM's capability in zero-shot multi-modal learning. Meanwhile, we also found that the unpretrained GPT-2 model outperforms the pretrained GPT-2 when condition-QoI examples are included in the prompts, i.e. few-shot learning. This may be attributed to that the pretrained language models are specialized in language understanding, while weak in numerical in-context operator learning. We leave this issue to future research.

6. Summary

We present a novel approach to transform in-context operator learning for scientific machine learning into a multi-modal framework, meaning that the model can learn the operator from captions, numerical function data, or a combination of both. These captions incorporate human knowledge about the operator, in the form of natural language descriptions and equations. We also introduce a more efficient model architecture for multi-modal in-context operator learning, namely "ICON-LM". This architecture closely aligns with language models, with carefully designed input sequences and transformer masks.

In the experiments, we compared the ICON-LM model with the baseline encoder–decoder ICON model, in the single-modal learning scenario. The proposed ICON-LM model, comprising approximately half the number of parameters, surpasses the performance of the baseline ICON model with less training time. We also showed the advantage of ICON-LM for in-context operator learning compared with FNO and DeepONet for classic operator learning, especially when the data is limited for the operators.

We also fine-tuned GPT-2 model for multi-modal in-context operator learning. We found that captions' presence, especially precise ones that disclose the parameters in the operators, significantly improved learning performance when the number of examples was limited. The model performance is improved even with vague captions that only disclose the operator type, showing impressive capabilities of the ICON-LM framework in combining multi-modal information to learn the operator.

In this paper, we focused on the transition to the multi-modal framework and the autoregressive training scheme, with the function representation limited to scattered points. Since transformers suffer from quadratic complexity in the number of input tokens, the current method may not be suitable in cases with dense data points, for example, fluid dynamics with grid data. However, the proposed multi-modal framework and "next function prediction" autoregressive training scheme should be able to generalize to other function representations, which are left to future work.

CRediT authorship contribution statement

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

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

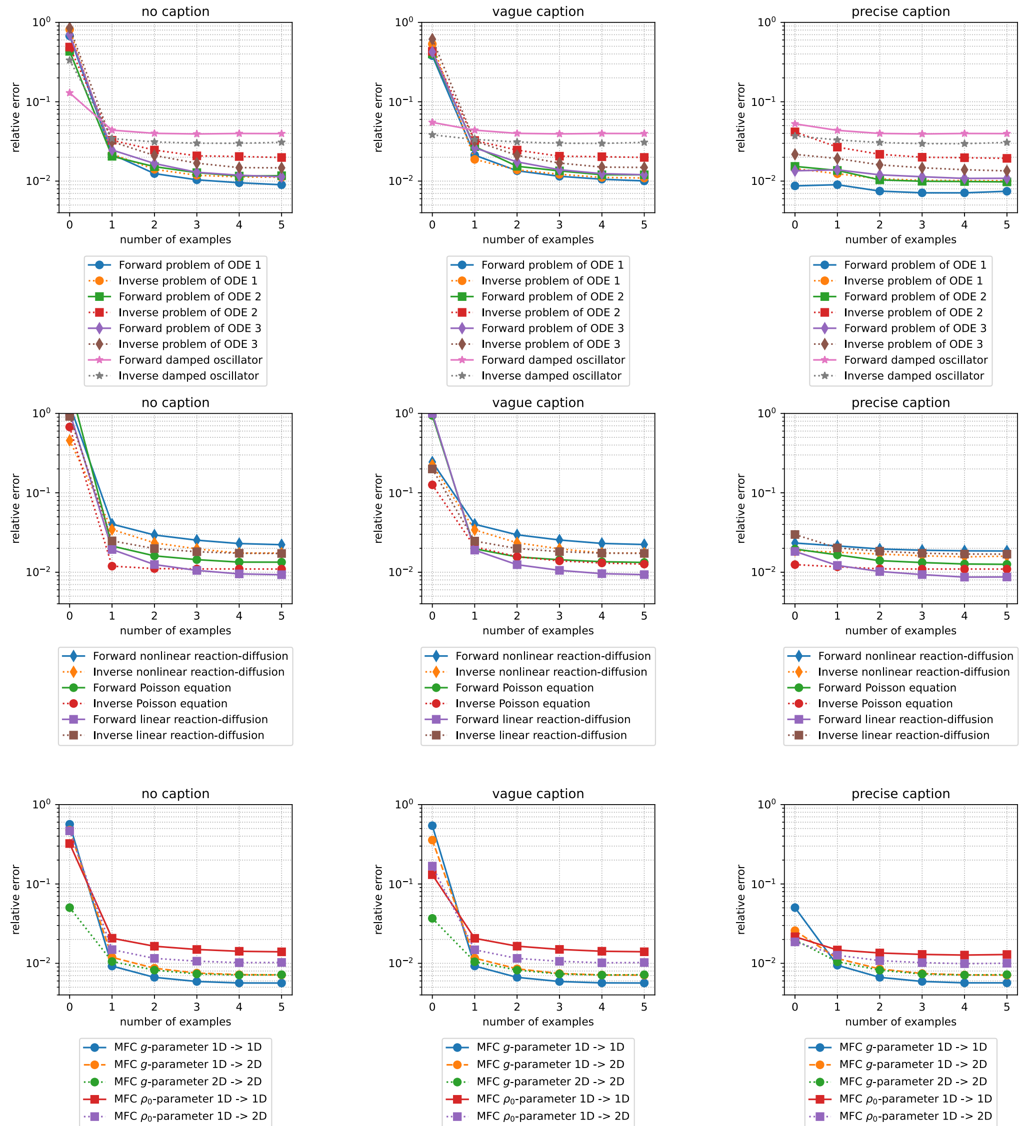
During the preparation of this work, the authors used ChatGPT to improve the expression and readability of this work. After using this tool, the authors reviewed and edited the content as needed and take full responsibility for the content of the publication.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

Siting Liu and Stanley J. Osher are supported by AFOSR MURI FA9550-18-502 and ONR MURI N00014-20-1-2787. We want to thank Dr. Tingwei Meng and Dr. Hedi Xia for the helpful discussions.



(a) (b) (c)

Fig. 9. Relative testing error for cases from zero-shot to five-shot learning, for each type of problem. (a) Testing without captions. (b) Testing with vague captions. (c) Testing with precise captions.

Appendix A. Caption data generation

We prompted the GPT-4 model to generate caption data, where the prompt consists of PDE descriptions in human language and LaTeX, instructions for captions, and several handcrafted caption examples. Most of the captions were suitable, though a few required slight manual adjustments. After proper post-processing, these captions were fed into the ICON-LM model.

We have two groups of captions. For the vague group, we instructed GPT-4 to use natural language to describe the equation or indicate its form without revealing the specific parameter values of the operators. For the precise group, we instructed GPT-4 to leave placeholders for the actual parameters, which were replaced with the actual parameter values during the post-processing stage. It is worth noting that for mean-field control problems, where the parameters are functional, we adopted a discretization approach to represent parameters.

Here, we present an example prompt for GPT-4 to generate vague and precise captions for ODE 1. The GPT-4 needs to be called multiple times to generate a sufficient number of captions.

```
Generate captions for an ordinary differential equation (
    ODE).
---
Here are some example captions:
An ODE with a state variable $u$ and control variable $c$.
$du(t)/dt = a \\cdot c(t) + b$.
An ODE $du(t)/dt = a \\cdot c(t) + 0.002$, with $a = 0.001$.
$du(t)/dt = b + a \\cdot c(t) $,  where $a = 0.001$ and $b =
    0.002$.
---
Now please design two groups of captions based on the above
     examples.
---
In the group 1, you can use human language or tell the form
    of the equation with parameters, but do not tell the
    value of the parameters. For example:
An ODE with a state variable $u$ and control variable $c$.
Function $u$ change over time with the rate of $a \cdot c(t
    ) + b$.
```

```
The differential equations is $du(t)/dt = a \cdot c(t) + b$.
---
In the group 2, you should tell all the values of the parameters $a$ and $b$. We give that $a = 0.001$, $b = 0.002.$ In the expression, you should specify the parameters values. The following examples are good:
$du(t)/dt = a \cdot c(t) + b$, where $b = 0.002, a = 0.001$.
The differential equations is $du(t)/dt = 0.001 \cdot c(t) + 0.002$.
---
Requirements:
0, You are an expert in the field of PDEs and ODEs. You have several publications in peer-reviewed journals. You are familiar with the notations and the equations.
1, In all examples, you should introduce the notations $u(t)$ and $c(t)$.
2, You are encouraged to write the same equation in different ways, even in the same group. For example, you can either use $du(t)/dt$ or $\frac{du(t)}{dt}$ to represent the time derivative of $u$.
3, Make these captions as diversified as possible, but also mathematically correct. You can reuse the example provided.
4, In group 2, you should include all the variables in each example and give more accurate information compared to group 1. Do not write ``the parameter needs to be determined'' or similar sentences.
5, group 1 should contain no specific values for the parameters.
---
In each group, using one line for each example. Do not use any format. Do not number them or use lists. Do not write ``group 1'' or related words. Do not use quotes. The answer should only contain the examples and the empty lines. Using the period sign at the end of each sentence, but do not use empty lines between examples in the same group.
---
Design and list all 20 examples for group 1 and 20 examples for group 2; First you list all 20 examples of group 1, then use one empty line to separate group 1 and group 2, next list all 20 examples for group 2.
```

Appendix B. Caption examples

Here we show several examples of training and testing captions for three characteristic problem types: ODE 3 forward problem, PDE 3 forward problem, and MFC g-parameter 1D → 1D. For each type, we show four vague captions and four precise captions, with the former two used in training, and the latter two for testing.

1. Caption examples for ODE 3 forward problem.

```
----Vague----
Variable $u$'s time derivative is $du(t)/dt = a_1 \cdot u(t) + a_2 \cdot c(t) + a_3$. Condition: $u(0)$ and $c(t), t\in[0,1]$, QoI: $u(t), t\in[0,1]$.
The ordinary differential equation represents the growth rate of variable $u(t)$ in relation to itself and the control function $c(t)$. Condition: $u(0)$ and $c(t), t\in[0,1]$, QoI: $u(t), t\in[0,1]$.
Derivation of $u(t)$ in time following the formula $du/dt = a_1u(t) + a_2c(t) + a_3$. Condition: $u(0)$ and $c(t), t\in[0,1]$, QoI: $u(t), t\in[0,1]$.
An ordinary differential equation with respect to time using a state variable $u(t)$ and a control variable $c(t)$. Condition: $u(0)$ and $c(t), t\in[0,1]$, QoI: $u(t), t\in[0,1]$.

----Precise----
Knowing that $a_1 = -0.0124, a_2 = 1.06, a_3 = 0.105$, the derivative $du(t)/dt = -0.0124 \cdot u(t) + 1.06 * c(t) + 0.105$. Condition: $u(0)$ and $c(t), t\in[0,1]$, QoI: $u(t), t\in[0,1]$.
The state variable changes according to $du(t)/dt = 0.347 \cdot u(t) + 0.535 \cdot c(t) + 0.459$. Condition: $u(0)$ and $c(t), t\in[0,1]$, QoI: $u(t), t\in[0,1]$.
Express an ODE as $\frac{du(t)}{dt} = -0.85 \cdot u(t) + 1.13 \cdot c(t) + -0.779$. Condition: $u(0)$ and $c(t), t\in[0,1]$, QoI: $u(t), t\in[0,1]$.
This differential equation $du(t)/dt = 0.167 * u(t) + 1.02 * c(t) + 0.457$ shows how $u(t)$ changes with time. Condition: $u(0)$ and $c(t), t\in[0,1]$, QoI: $u(t), t\in[0,1]$.
```

2. Caption examples for PDE 3 forward problem.

```
----Vague----
The nonlinear PDE, written as $-\lambda\frac{d^2u}{dx^2} + a * u^3 = c(x)$, includes the variables $u(x)$ and $c(x)$. Condition: $c(x), x\in[0,1]$, QoI: $u(x), x\in[0,1]$.
The nonlinear PDE, $u''(x) - a \cdot u(x)^3 = c(x)$ roping in $u(x)$ and $c(x)$. Condition: $c(x), x\in[0,1]$, QoI: $u(x), x\in[0,1]$.
Ponder upon this nonlinear PDE, involving the dependent variable $u(x)$ and the term $c(x)$ constituting the source. Condition: $c(x), x\in[0,1]$, QoI: $u(x), x\in[0,1]$.
This PDE, $-\lambda d^2u/dx^2 + a \cdot u^3 = c(x)$, involves the variables $u(x)$ and $c(x)$. Condition: $c(x), x\in[0,1]$, QoI: $u(x), x\in[0,1]$.

----Precise----
For the given equation $- 0.101 * \frac{d^2u}{dx^2} + 1.16 * u^3 = c(x)$, we have the boundary conditions $u(0) = -0.517$ and $u(1) = -0.689$. Condition: $c(x), x\in[0,1]$, QoI: $u(x), x\in[0,1]$.
The nonlinear PDE is $-0.0504 d^2u/dx^2 + 0.705 \cdot u^3 = c(x)$, with $u(0) = -0.319$ and $u(1) = -0.667$. Condition: $c(x), x\in[0,1]$, QoI: $u(x), x\in[0,1]$.
Equation $- 0.116 \frac{d^2u}{dx^2} + 0.586 \cdot u^3 = c(x)$, is our PDE with $u(0) = 0.322$ and $u(1) = -0.749$. Condition: $c(x), x\in[0,1]$, QoI: $u(x), x\in[0,1]$.
Let us examine this PDE $- 0.139 \frac{d^2u}{dx^2} + 1.25 \cdot u^3 = c(x)$, imposing $u(0) = -0.351$, $u(1) = 0.597$. Condition: $c(x), x\in[0,1]$, QoI: $u(x), x\in[0,1]$.
```

3. Caption examples for MFC g-parameter 1D → 1D.

```
----Vague----
Investigating Mean Field Control Problem involving an interplay between density $\rho$ and an uncertain function $g$ inside the terminal cost. Condition: $\rho(0,x), x\in[0,1]$, QoI: $\rho(1,x), x\in[0,1]$.
In the mean field control problem, we minimize $\int \int \frac{10m^2}{\rho} dx dt + \int g(x)\rho(1,x) dx$, subject to $\partial_{t}\rho + \nabla_{x}\cdot m = 0.02 \Delta_{x}\rho$, with $\rho(0,x)=\rho_{0}(x)$, where $g$ is an unknown function. Condition: $\rho(0,x), x\in[0,1]$, QoI: $\rho(1,x), x\in[0,1]$.
```

```
Consider the mean field control problem with density function $\rho(t,x)$ and terminal cost $\int g(x)\rho(1,x) dx$ where $g$ is an unknown function. Condition: $\rho(0,x), x\in[0,1]$, QoI: $\rho(1,x), x\in[0,1]$.
The mean field control problem formulates $\inf_{\rho, m}\iint \frac{10m^2}{\rho} dx dt + \int g(x)\rho(1,x) dx$ subject to $\partial_t \rho(t,x) + \nabla_x m(t,x) = 0.02 \Delta_x \rho(t,x)$, where $g(x)$ is undefined. Condition: $\rho(0,x), x\in[0,1]$, QoI: $\rho(1,x), x\in[0,1]$.

----Precise----
The analysis of $\inf_{\rho, m}\iint \frac{10m^2}{\rho} dx dt + \int g(x)\rho(1,x) dx$ for $t \in [0,1]$, $x \in [0,1]$ and periodic spatial boundary condition, under constraint of $\partial_t \rho(t,x) + \nabla_x m(t,x) = 0.02 \Delta_x \rho(t,x)$, with the function $g$ acting as terminal cost is defined as $g(0), g(0.1), ..., g(0.9)$ = 0.903, 0.957, 0.459, -0.178, -0.83, -1.5, -1.39, -0.189, 0.857, 0.909. Condition: $\rho(0,x), x\in[0,1]$, QoI: $\rho(1,x), x\in[0,1]$.
Analyzing mean field control problem $\inf_{\rho, m}\iint \frac{10m^2}{\rho} dx dt + \int g(x)\rho(1,x) dx$, subject to the constraints, for $t \in [0,1], x \in [0,1], and terminal function $g$ defined as $g(0), g(0.1), ..., g(0.9) = -0.244, 0.326, 0.598, 0.571, 0.287, 0.0734, 0.00921, -0.299, -0.67, -0.652$. Condition: $\rho(0,x), x\in[0,1]$, QoI: $\rho(1,x), x\in[0,1]$.
We solve a mean field control problem that seeks to minimize $\inf_{\rho, m}\iint \frac{10m^2}{\rho} dx dt + \int g(x)\rho(1,x) dx$ while adhering to $\partial_t \rho(t,x) + \nabla_x m(t,x) = 0.02 \Delta_x \rho(t,x)$ and $\rho(0,x) =\rho_0(x)$. A known function $g$ is given by $g(0), g(0.1), ..., g(0.9)$ = 0.535, 0.976, 1.35, 1.49, 0.135, -1.86, -1.66, -0.692, -0.305, 0.0242. Condition: $\rho(0,x), x\in[0,1]$, QoI: $\rho(1,x), x\in[0,1]$.
Studying the mean field control problem $\inf_{\rho, m}\iint \frac{10m^2}{\rho} dx dt + \int g(x)\rho(1,x) dx$, where $t \in [0,1], x \in [0,1], and $g$ is given as $g(0), g(0.1), ..., g(0.9) = -0.0268, 0.196, 0.08, -0.0463, 0.145, 0.126, 0.169, 0.0845, -0.313, -0.413$. Condition: $\rho(0,x), x\in[0,1]$, QoI: $\rho(1,x), x\in[0,1]$.
```

Appendix C. Neural network and training configurations

The transformer used in Section 5.1 is configured as in Table C.4. Both the embedding layer and head layer are linear layers. For fine-tuning the GPT-2 model in Section 5.3, we apply shallow multilayer perceptrons as the input embedding layer for function data as well as the output head layer, with one hidden layer of dimension 1024. We utilize the AdamW optimizer with a warmup-cosine-decay schedule, employing the configuration in Table C.5.

In Section 5.1, we pretrained and fine-tuned the FNO and DeepONet. The FNO model is adopted from the official implementation (https://github.com/neuraloperator/neuraloperator), with `n_modes` = 16, `hidden_channels` = 512, `in_channels` = 2 (one for x, one for $u(x)$), and `out_channels` = 1. Other parameters are default. The total number of trainable parameters is about 20.2 million. In the DeepONet, both the trunk net and branch net are 6-layer multilayer perceptrons, with hidden and output widths of 1024. The total number of trainable parameters is about 14.8 million. The pretraining of FNO and DeepONet has a batch size of 32, following the same configuration as in Table C.5, except that the total training steps are reduced to 0.1 million, which is sufficient for convergence. As for fine-tuning FNO and DeepONet, we use all the available five examples in the training batch, employing the AdamW optimizer with a constant learning rate 1e-5, weight decay 1e-4, and global norm clip 1.0.

Table C.4
Configuration of the transformer in single-modal ICON-LM.

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
Total Trainable Parameters	15.8 Million

Table C.5
Configuration of optimizer and learning rate schedule.

Initial learning rate	0.0
Peak Learning Rate	1e−4
End Learning Rate	0.0
Total Training Steps	1 million
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

Data availability

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

References

Alayrac, J. B., Donahue, J., Luc, P., Miech, A., Barr, I., Hasson, Y., Lenc, K., Mensch, A., Millican, K., Reynolds, M., et al. (2022). Flamingo: A visual language model for few-shot learning. *Advances in Neural Information Processing Systems*, *35*, 23716–23736.

Bhattacharya, K., Hosseini, B., Kovachki, N. B., & Stuart, A. M. (2021). Model reduction and neural networks for parametric PDEs. *The SMAI Journal of Computational Mathematics*, *7*, 121–157.

Bommasani, R., Hudson, D. A., Adeli, E., Altman, R., Arora, S., von Arx, S., Bernstein, M. S., Bohg, J., Bosselut, A., Brunskill, E., et al. (2021). On the opportunities and risks of foundation models. arXiv preprint arXiv:2108.07258.

Brohan, A., Brown, N., Carbajal, J., Chebotar, Y., Chen, X., Choromanski, K., Ding, T., Driess, D., Dubey, A., Finn, C., Florence, P., Fu, C., Arenas, M. G., Gopalakrishnan, K., Han, K., Hausman, K., Herzog, A., Hsu, J., Ichter, B., ..., Irpan, A. et al. (2023). RT-2: Vision-language-action models transfer web knowledge to robotic control.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. (2020). Language models are few-shot learners. *Advances in Neural Information Processing Systems*, *33*, 1877–1901.

Chen, T., & Chen, H. (1995a). Approximation capability to functions of several variables, nonlinear functionals, and operators by radial basis function neural networks. *IEEE Transactions on Neural Networks*, *6*(4), 904–910.

Chen, T., & Chen, H. (1995b). Universal approximation to nonlinear operators by neural networks with arbitrary activation functions and its application to dynamical systems. *IEEE Transactions on Neural Networks*, *6*(4), 911–917.

Driess, D., Xia, F., Sajjadi, M. S., Lynch, C., Chowdhery, A., Ichter, B., Wahid, A., Tompson, J., Vuong, Q., Yu, T., et al. (2023). PaLM-E: An embodied multimodal language model. arXiv:2303.03378.

E, W., Han, J., & Jentzen, A. (2017). Deep learning-based numerical methods for high-dimensional parabolic partial differential equations and backward stochastic differential equations. *Communications in Mathematics and Statistics*, *5*(4), 349–380.

E, W., & Yu, B. (2018). The deep Ritz method: A deep learning-based numerical algorithm for solving variational problems. *Commun. Math. Stat.*, *6*(1), 1–12. http://dx.doi.org/10.1007/s40304-018-0127-z.

Gao, P., Han, J., Zhang, R., Lin, Z., Geng, S., Zhou, A., Zhang, W., Lu, P., He, C., Yue, X., et al. (2023). LLaMA-adapter V2: Parameter-efficient visual instruction model. arXiv preprint arXiv:2304.15010.

Goswami, S., Kontolati, K., Shields, M. D., & Karniadakis, G. E. (2022). Deep transfer operator learning for partial differential equations under conditional shift. *Nature Machine Intelligence*, 1–10.

Han, J., Jentzen, A., & E, W. (2018). Solving high-dimensional partial differential equations using deep learning. *Proceedings of the National Academy of Sciences of the United States of America*, *115*(34), 8505–8510. http://dx.doi.org/10.1073/pnas.1718942115.

Jin, P., Zhang, Z., Zhu, A., Tang, Y., & Karniadakis, G. E. (2020). SympNets: Intrinsic structure-preserving symplectic networks for identifying Hamiltonian systems. *Neural Networks*, *132*, 166–179.

Karniadakis, G. E., Kevrekidis, I. G., Lu, L., Perdikaris, P., Wang, S., & Yang, L. (2021). Physics-informed machine learning. *Nature Reviews Physics*, *3*(6), 422–440.

Khoo, Y., Lu, J., & Ying, L. (2021). Solving parametric PDE problems with artificial neural networks. *European Journal of Applied Mathematics*, *32*(3), 421–435.

Kissas, G., Seidman, J. H., Guilhoto, L. F., Preciado, V. M., Pappas, G. J., & Perdikaris, P. (2022). Learning operators with coupled attention. *Journal of Machine Learning Research*, *23*(215), 1–63.

Kochkov, D., Smith, J. A., Alieva, A., Wang, Q., Brenner, M. P., & Hoyer, S. (2021). Machine learning–accelerated computational fluid dynamics. *Proceedings of the National Academy of Sciences*, *118*(21), Article e2101784118.

Kovachki, N., Li, Z., Liu, B., Azizzadenesheli, K., Bhattacharya, K., Stuart, A., & Anandkumar, A. (2023). Neural operator: Learning maps between function spaces with applications to PDEs. *Journal of Machine Learning Research*, *24*(89), 1–97.

Li, K., He, Y., Wang, Y., Li, Y., Wang, W., Luo, P., Wang, Y., Wang, L., & Qiao, Y. (2023). VideoChat: Chat-centric video understanding. arXiv preprint arXiv:2305.06355.

Li, Z., Kovachki, N. B., Azizzadenesheli, K., liu, B., Bhattacharya, K., Stuart, A., & Anandkumar, A. (2021). Fourier neural operator for parametric partial differential equations. In *International conference on learning representations*. URL: https://openreview.net/forum?id=c8P9NQVtmnO.

Li, J., Li, D., Savarese, S., & Hoi, S. (2023). BLIP-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. arXiv preprint arXiv:2301.12597.

Li, Z., Zheng, H., Kovachki, N., Jin, D., Chen, H., Liu, B., Azizzadenesheli, K., & Anandkumar, A. (2021). Physics-informed neural operator for learning partial differential equations. arXiv preprint arXiv:2111.03794.

Ling, J., Kurzawski, A., & Templeton, J. (2016). Reynolds averaged turbulence modelling using deep neural networks with embedded invariance. *Journal of Fluid Mechanics*, *807*, 155–166.

Liu, J. W., Erichson, N. B., Bhatia, K., Mahoney, M. W., & Re, C. (2023). Does in-context operator learning generalize to domain-shifted settings? In *The symbiosis of deep learning and differential equations III*.

Liu, H., Li, C., Wu, Q., & Lee, Y. J. (2023). Visual instruction tuning. arXiv:2304.08485.

Liu, Y., Zhang, Z., & Schaeffer, H. (2023). Prose: Predicting operators and symbolic expressions using multimodal transformers. arXiv preprint arXiv:2309.16816.

Long, Z., Lu, Y., Ma, X., & Dong, B. (2018). PDE-Net: Learning PDEs from data. In *International conference on machine learning* (pp. 3208–3216). PMLR.

Lu, L., Jin, P., Pang, G., Zhang, Z., & Karniadakis, G. E. (2021). Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators. *Nature Machine Intelligence*, *3*(3), 218–229.

Lusch, B., Kutz, J. N., & Brunton, S. L. (2018). Deep learning for universal linear embeddings of nonlinear dynamics. *Nature Communications*, *9*(1), 4950.

Mattheakis, M., Protopapas, P., Sondak, D., Di Giovanni, M., & Kaxiras, E. (2019). Physical symmetries embedded in neural networks. arXiv preprint arXiv:1904.08991.

Pfau, D., Spencer, J. S., Matthews, A. G., & Foulkes, W. M. C. (2020). Ab initio solution of the many-electron Schrödinger equation with deep neural networks. *Physical Review Research*, *2*(3), Article 033429.

Pi, R., Gao, J., Diao, S., Pan, R., Dong, H., Zhang, J., Yao, L., Han, J., Xu, H., & Zhang, L. K. T. (2023). DetGPT: Detect what you need via reasoning. arXiv preprint arXiv:2305.14167.

Pun, G. P., Batra, R., Ramprasad, R., & Mishin, Y. (2019). Physically informed artificial neural networks for atomistic modeling of materials. *Nature Communications*, *10*(1), 2339.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I., et al. (2019). Language models are unsupervised multitask learners. *OpenAI Blog*, *1*(8), 9.

Raissi, M., Perdikaris, P., & Karniadakis, G. E. (2019). Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, *378*, 686–707. http://dx.doi.org/10.1016/j.jcp.2018.10.045.

Ruthotto, L., Osher, S. J., Li, W., Nurbekyan, L., & Fung, S. W. (2020). A machine learning framework for solving high-dimensional mean field game and mean field control problems. *Proceedings of the National Academy of Sciences of the United States of America*, *117*(17), 9183–9193. http://dx.doi.org/10.1073/pnas.1922204117.

Sirignano, J., & Spiliopoulos, K. (2018). DGM: A deep learning algorithm for solving partial differential equations. *Journal of Computational Physics*, *375*, 1339–1364. http://dx.doi.org/10.1016/j.jcp.2018.08.029.

Subel, A., Guan, Y., Chattopadhyay, A., & Hassanzadeh, P. (2023). Explaining the physics of transfer learning in data-driven turbulence modeling. *PNAS Nexus*, *2*(3), pgad015.

Subramanian, S., Harrington, P., Keutzer, K., Bhimji, W., Morozov, D., Mahoney, M., & Gholami, A. (2023). Towards foundation models for scientific machine learning: Characterizing scaling and transfer behavior. arXiv preprint arXiv:2306.00258.

Tsimpoukelli, M., Menick, J. L., Cabi, S., Eslami, S., Vinyals, O., & Hill, F. (2021). Multimodal few-shot learning with frozen language models. *Advances in Neural Information Processing Systems*, *34*, 200–212.

Wang, S., Wang, H., & Perdikaris, P. (2021). Learning the solution operator of parametric partial differential equations with physics-informed DeepOnets. *Science Advances*, *7*(40), eabi8605.

Yang, Z., Gan, Z., Wang, J., Hu, X., Lu, Y., Liu, Z., & Wang, L. (2022). An empirical study of GPT-3 for few-shot knowledge-based VQA. In *Proceedings of the AAAI conference on artificial intelligence*: *vol. 36, no. 3*, (pp. 3081–3089).

Yang, L., Liu, S., Meng, T., & Osher, S. J. (2023). In-context operator learning with data prompts for differential equation problems. *Proceedings of the National Academy of Sciences*, *120*(39), Article e2310142120.

Yang, L., & Osher, S. J. (2024). PDE generalization of in-context operator networks: A study on 1D scalar nonlinear conservation laws. arXiv preprint arXiv:2401.07364.

Ye, Z., Huang, X., Chen, L., Liu, H., Wang, Z., & Dong, B. (2024). PDEformer: Towards a foundation model for one-dimensional partial differential equations. arXiv preprint arXiv:2402.12652.

Yin, S., Fu, C., Zhao, S., Li, K., Sun, X., Xu, T., & Chen, E. (2023). A survey on multimodal large language models. arXiv preprint arXiv:2306.13549.

Zang, Y., Bao, G., Ye, X., & Zhou, H. (2020). Weak adversarial networks for high-dimensional partial differential equations. *Journal of Computational Physics*, *411*, 109409, 14. http://dx.doi.org/10.1016/j.jcp.2020.109409.

Zhang, L., Han, J., Wang, H., Car, R., & Weinan, E. (2018). Deep potential molecular dynamics: A scalable model with the accuracy of quantum mechanics. *Physical Review Letters*, *120*(14), Article 143001.

Zhang, R., Han, J., Zhou, A., Hu, X., Yan, S., Lu, P., Li, H., Gao, P., & Qiao, Y. (2023). LLaMA-adapter: Efficient fine-tuning of language models with zero-init attention. arXiv preprint arXiv:2303.16199.

Zhang, H., Li, X., & Bing, L. (2023). Video-LLaMA: An instruction-tuned audio-visual language model for video understanding. arXiv preprint arXiv:2306.02858.

Zhang, X., Wu, C., Zhao, Z., Lin, W., Zhang, Y., Wang, Y., & Xie, W. (2023). PMC-VQA: Visual instruction tuning for medical visual question answering. arXiv preprint arXiv:2305.10415.

Zhu, D., Chen, J., Shen, X., Li, X., & Elhoseiny, M. (2023). MiniGPT-4: Enhancing vision-language understanding with advanced large language models. arXiv preprint arXiv:2304.10592.

Zhu, Y., & Zabaras, N. (2018). Bayesian deep convolutional encoder–decoder networks for surrogate modeling and uncertainty quantification. *Journal of Computational Physics*, *366*, 415–447.

Zhu, M., Zhang, H., Jiao, A., Karniadakis, G. E., & Lu, L. (2023). Reliable extrapolation of deep neural operators informed by physics or sparse observations. *Computer Methods in Applied Mechanics and Engineering*, *412*, Article 116064.