Latent Neural Operator for Solving Forward and Inverse PDE Problems

Tian Wang
State Key Laboratory of
Multimodal Artificial Intelligence Systems
Institute of Automation,
Chinese Academy of Sciences
School of Artificial Intelligence,
University of Chinese Academy of Sciences
wangtian2022@ia.ac.cn

Chuang Wang*
State Key Laboratory of
Multimodal Artificial Intelligence Systems
Institute of Automation,
Chinese Academy of Sciences
School of Artificial Intelligence,
University of Chinese Academy of Sciences
wangchuang@ia.ac.cn

Abstract

Neural operators effectively solve PDE problems from data without knowing the explicit equations, which learn the map from the input sequences of observed samples to the predicted values. Most existing works build the model in the original geometric space, leading to high computational costs when the number of sample points is large. We present the Latent Neural Operator (LNO) solving PDEs in the latent space. In particular, we first propose Physics-Cross-Attention (PhCA) transforming representation from the geometric space to the latent space, then learn the operator in the latent space, and finally recover the real-world geometric space via the inverse PhCA map. Our model retains flexibility that can decode values in any position not limited to locations defined in the training set, and therefore can naturally perform interpolation and extrapolation tasks particularly useful for inverse problems. Moreover, the proposed LNO improves both prediction accuracy and computational efficiency. Experiments show that LNO reduces the GPU memory by 50%, speeds up training 1.8 times, and reaches state-of-the-art accuracy on four out of six benchmarks for forward problems and a benchmark for inverse problem. Code is available at https://github.com/L-I-M-I-T/LatentNeuralOperator.

1 Introduction

Neural network approaches for partial differential equations (PDEs) attract increasing attention in diverse scientific fields, for instance, meteorological prediction[1], industrial design[2], geological sensing[3] and environmental monitoring[4] to name a few. Compared with traditional numerical methods, *e.g.*, finite element method[5], which demands a large number of computational resources and requires meticulous discretization involving a lot of specialized knowledge, neural networks provide a new paradigm for solving PDE-driven problems with a better trade-off between accuracy and flexibility.

Neural PDE solvers can be divided into two main categories, physics-informed and operator-learning models. The physics-informed methods[6–11] enforce the inductive bias of explicit PDE instances into loss function or model architecture, usually yielding solutions with high accuracy, but are less flexible as they require fresh training to find the solution for each new instance. In contrast, the operator learning methods[12–15] adopt a data-driven paradigm to learn the implicit PDE constraint

*Corresponding Author.This work is supported by Beijing Municipal Science and Technology Project (No. Z231100010323005), the 2035 Innovation Mission (Grants No. E4J10102) and Pioneer Hundred Talents Program of CAS under Grant Y9S9MS08 and E2S40101.

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

https://doi.org/10.52202/079017-1042

from pair of samples representing input-to-output functional mappings, providing better flexibility that can generalize to different initial/boundary conditions beyond the scope of the training data, but have yet large room to improve in the prediction accuracy.

Recently, Transformers[16] prevail in neural operator structures. Theoretically, the attention mechanism was proved to be a special form of the integral kernel[15, 17] in neural operator, which naturally conforms to sequence-to-sequence characterization. Moreover, the fully-connected attention structure enables the model to characterize long-distance and quadratic relationships among sample points, which yields more accurate prediction than the vanilla MLP structure[13] but at the expense of increasing the computational complexity drastically, as the complexity is quadratic respect to the length of input sequence.

To alleviate the complexity, existing works employed linear attention mechanism [18–21] speeding up the computation but sacrificing precision due to their limited expressive power. Another line of works attempted to solve the PDE in the latent space with a small number of samples [22–26], which get rid of abundant sample points in the original geometric space and capture physical correlations in the tighter latent space.

In this work, we present the Latent Neural Operator (LNO) with particular effort on designing the **Ph**ysics-**C**ross-**A**ttension (PhCA) module to learn the latent space to optimize the trade-off between accuracy and flexibility, and reduce the computational complexity as well. Specifically, LNO first encodes the input sample sequence into a learnable latent space by PhCA. Then, the model learns the PDE operator in the latent space via a stack of Transformer layers. Finally, the predicted output function is decoded by the inverse PhCA, given any query locations. Unlike previous works that either pre-define the latent space as frequency domain [14], or switch back-and-forth between latent space and geometric space at each layer [23], the latent space of the LNO is learnable from data and the data are only transformed at the first and last layer.

The proposed PhCA module decouples the locations of input samples fed in the encoder and output samples fed in the decoder, which can predict values in any position not limited by the observation. Therefore, it can naturally perform interpolation and extrapolation tasks, which is fundamental for solving inverse problems. The proposed LNO improves both prediction accuracy and computational efficiency. Experiments show that LNO reduces the GPU memory by 50% and speeds up training 1.8 times. Moreover, it reaches state-of-the-art accuracy on four (Pipe, Elasticity, Darcy, Navier-Stokes) out of six benchmarks for forward problems and a benchmark for inverse problem on Burgers' equation.

The main contributions of our LNO framework are summarized as follows.

- Flexibility: We propose the Physics-Cross-Attention (PhCA) module which decouples the locations of input and output samples and learns the latent space from data. We also build the Latent Neural Operator (LNO) model for solving both forward and inverse PDE problems.
- Efficiency: The LNO model transforms the data to the latent space only once. Compared with other approaches that switch the latent space and geometric space in each Transformer layer, the proposed model reduces complexity drastically in terms of GPU memory usage, training time and number of model parameters.
- Accuracy: We obtain state-of-the-art results on the forward problems of Pipe, Elasticity, Darcy, Navier-Stokes, and the inverse problem of Burgers' equation.

2 Related works

We first review two major classes of neural PDE approaches, physics-informed and operator-learning methods. Then, we briefly discuss the works using neural networks for solving inverse PDE problems.

2.1 Physics-Informed Machine Learning

Physics-informed machine learning methods incorporate prior knowledge of PDEs into loss function or model architecture. The Physics-Informed Neural Network (PINN)[7] and its variants are among the most classic methods. PINN integrates the differential equation, initial conditions and boundary conditions into the loss function, guiding the neural network to approximate the solution

by ensuring its derivatives satisfies the equation and the output aligns with the initial and boundary conditions. Other methods attempt to encode physical priors into the model architecture, *e.g.,* the FInite volume Neural Network (FINN)[8] and Physics-encoded Recurrent Convolutional Neural Network (PeRCNN)[10]. These methods achieve high precision but require solving optimization to train the networks for each instance. In addition, in the case, where concrete PDE is unknown or not exact, it is challenging to adopt those methods, limiting the generalization ability and flexibility in practical data-rich situations.

2.2 Operator Learning

Operator learning methods aim to find a functional map from coefficients, forcing terms and initial conditions to the solution function by learning an approximating operator. These methods train neural networks to capture the correspondence between input and output functions from data without needing to know the underlying PDE explicitly. DeepONet[13, 27] first introduces approximating operators using neural networks.

A series of kernel integral operator forms were proposed to model the mappings between functions as neural operators[28, 14, 15]. Specifically, Galerkin Transformer[20] utilizes the Galerkin-type attention mechanism as kernel integral operators. OFormer[29] improves Galerkin-type attention by decoupling observation positions and query positions through cross-attention. GNOT[21] introduces heterogeneous normalized cross-attention to modulate various input conditions for query positions and alleviates computational pressure of the large grid by linear-time-complexity attention mechanism. FactFormer[30] proposes axial factorized kernel integral, decomposing input functions into multiple one-dimensional sub-functions, effectively reducing computation. ONO[31] enhances the generalization of neural operators by adding orthogonality regularization in attention mechanisms.

Researchers also explored the idea of modeling in the latent space. Transolver[23] employs Physics-Attention in each layer to map back-and-forth between geometric features and physical features. LSM[22] encodes the input function into the latent space using cross-attention, constructs a set of orthogonal bases, and then decodes the solution back to the geometric space through cross-attention again. UPT[24] introduces encoding and decoding losses, and compresses geometric point information into super-node information in the latent space via graph neural network[32].

Compared with existing methods, we autonomously learn mappings between functions in the real-world space and the latent space in an end-to-end fashion with the cross-attention mechanism, without manually constructed orthogonal basis nor additionally introduced loss.

2.3 Inverse Problem

Inverse problems of PDEs have extensive applications in fields involving sensing and reconstruction such as medical imaging and geological sensing. For instance, Transformer-based deep direct sampling methods[33] have been proposed for electrical impedance tomography, and convolutional network methods[34–36] are widely used in full waveform inversion.

Theoretical works on solving inverse problems have also been proposed. The reconstruction of diffusion fields with localized sources has been studied[37] from the perspective of sampling matrix in the spatiotemporal domain. PINN[38–40] approximates the solution based on partially observed data to compute the unknown data. NIO[41] combines DeepONet[13] and FNO[14] to construct a framework for solving inverse problems. FINN[8, 42] sets boundary conditions as learnable parameters, and infers boundary values through backpropagation. The generative method[43] generates the initial conditions through latent codes and refines them based on the results of time evolution. We aim to unify the solution of forward and inverse problems in the latent space through the PhCA module, which decouples observation and prediction positions.

3 Method

We formally define both the forward and the inverse problems of PDEs. Then, we introduce our Latent Neural Operator (LNO) model. Finally, we discuss the core module that learns the transformation between the real-world geometric space and the latent space via the cross-attention mechanism.

 https://doi.org/10.52202/079017-1042

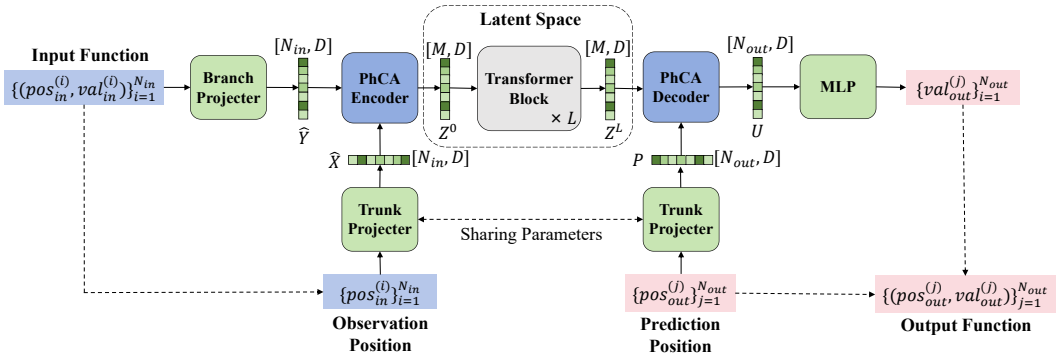


Figure 1: The overall architecture of Latent Neural Operator.

3.1 Problem Setup

We consider a partial differential equation (PDE) with boundary and/or initial conditions defined on $\Omega \subseteq \mathbb{R}^d$ or $\mathbb{R}^d \times [0, T)$

$$\begin{aligned} \mathcal{L}_a \circ u &= 0, \\ u(x) &= b(x), \quad x \in \partial\Omega \end{aligned} \tag{1}$$

where $\mathcal{L}_a$ is an operator containing partial differentials parameterized by a set of coefficients a, and $u(x)$, $x \in \Omega$ and $b(x)$ represent the solution to the PDE and the boundary/initial conditions respectively.

A conventional forward problem aims to find the solution $u(x)$, given the differential operator $\mathcal{L}_a$ in (1) with the explicit parameters a and the boundary/initial conditions $b(x)$. For an inverse problem, given an observed set of partial data $\{(x, u(x))|\ x \in \tilde{\Omega} \subset \Omega\}$, the task is to infer the PDE that generates the observed data along with possible boundary/initial conditions or the parameters a in the differential operator $\mathcal{L}_a$.

Both forward and inverse problems can be unified as an operator learning task fitting into a sequence-to-sequence translation framework. An operator model estimates the mapping $\mathcal{F} : f \mapsto g$ from an input function f to an output function g by the data-driven approach using paired training data of input-output sequences $(\{f(y_i)\}_i, \{g(z_j)\}_j)$, where y_i and z_j correspond to positions in the domain of the input and output function respectively.

Specifically, for the forward problem, the inputs are samples of either boundary/initial function b or PDE coefficient function a, and the output is the physical quantity u evaluated at a grid of points. Conversely, for the inverse problem, the input is a few partially observation samples $\{(pos_{in}^{(i)}, val_{in}^{(i)})\}_{i=1}^{N_{in}} = \{(x_i, u(x_i))\}_{i=1}^{N_{in}}$, and the outputs are boundary/initial function (*e.g.,* infer initial state) or the PDE coefficient function (also coined as system identification problem).

The challenge in solving forward and inverse problems of PDEs lies in the large lengths of input and output sequences. In terms of operator approximating, complex kernel integrals lead to high computational cost while simple kernel integrals lack accuracy. In this work, we build an accurate and computationally efficient model for both forward and inverse problems by learning the operator in the learnable latent space.

3.2 Latent Neural Operator

Overview The proposed model of Latent Neural Operator (LNO) is applicable for both forward and inverse PDE problems. It consists of four modules: an embedding layer to lift the dimension of the input data, an encoder to transform the input data to a learnable latent space, a series of self-attention layers for modeling operator in the latent space, and a decoder to recover the latent representation to the real-world geometric space. The overall architecture of LNO is shown in Figure 1.

Embedding inputs The geometric space is the original space of the PDE input or output which contains N_{in} or N_{out} samples, each with d-dimensional position coordinates (which may additionally

include a 1-dimensional time coordinate for time-dependent PDEs) and n-dimensional physical quantity values.

First, we embed the positions and physics quantity values of the input sequence in the geometric space to higher D-dimensional representations. This process is implemented by two MLPs, where the first only embeds the position information and the second embeds both the location and physics quantity information

$$\begin{aligned} \hat{x}^{(i)} &= \text{trunk-projector}\big(pos_{in}^{(i)}\big), && \hat{x}^{(i)} \in \mathbb{R}^{D\times 1} \\ \hat{y}^{(i)} &= \text{branch-projector}\big(\text{concat}(pos_{in}^{(i)}, val_{in}^{(i)})\big), && \hat{y}^{(i)} \in \mathbb{R}^{D\times 1}. \end{aligned}$$

Analogous to DeepONet [13], we name these MLPs as trunk-projector and branch-projector respectively. The embedding operation helps to map the locations and physics quantities to the embedding space where their relationships are easier to capture.

Removing the physics quantity values from the trunk-projector input grants our model the decoupling property. In this way, we can predict values at positions not included in the sampled input function from the latent space, thus achieving the decoupling of observation and prediction positions. Since the sampled input function always includes position-value pairs, using both position and value information in the branch projector will not affect the decoupling objective.

Encoding to the latent space We attempt to represent the input function using the representation tokens of M hypothetical sampling positions $\{h^{(k)}\}_{k=1}^{M}, h^{(k)} \in R^{D\times 1}$ which exist in the latent space, where M can be much smaller than the number N_{in} of samples of raw inputs.

We use the Physics-Cross-Attention to model the transformation from the geometric space to the latent space. The inputs are query $\boldsymbol{H} = [h^{(1)}, h^{(2)}..., h^{(M)}]^T$, key $\hat{\boldsymbol{X}} = [\hat{x}^{(1)}, \hat{x}^{(2)}, \ldots, \hat{x}^{(N_{in})}]^T$ and value $\hat{\boldsymbol{Y}} = [\hat{y}^{(1)}, \hat{y}^{(2)}, \ldots, \hat{y}^{(N_{in})}]^T$ which corresponding to embedding of hypothetical locations in the target latent space, embedding of locations in the source real-world geometric space, and embedding of the concatenation of locations and physics quantities in the source real-world geometric space respectively. The output can be calculated through

$$\boldsymbol{Z}^0 = \text{Physics-Cross-Attention}(\boldsymbol{H}, \hat{\boldsymbol{X}}, \hat{\boldsymbol{Y}}) \tag{2}$$

In contrast to FNO[14], where the latent space is pre-defined as the frequency domain, we learn the latent space from data. Therefore, the embeddings of sample locations $\boldsymbol{H}$ as queries in the cross-attention are also learnable parameters.

After encoding the input function into the latent space, the length of the sequence to be processed is reduced from N_{in} to M. With $M \ll N_{in}$, extracting and converting the feature of the input function in the latent space will be much more efficient than in the real-world geometric space.

Learning operator in the latent space We model the operator of the forward/inverse PDE problem in the latent space to convert the feature of the input function to that of the output function, using a stack of Transformer blocks with the self-attention mechanism

$$\boldsymbol{Z}^l = \text{MLP}(\text{LayerNorm}(\hat{\boldsymbol{Z}}^l)) + \hat{\boldsymbol{Z}}^l \quad \text{with } \hat{\boldsymbol{Z}}^l = \text{Attention}(\text{LayerNorm}(\boldsymbol{Z}^{l-1})) + \boldsymbol{Z}^{l-1}$$

where $l \in \{1, 2, ..., L\}$ is the index of Transformer block with L being the total number of the blocks. We experiment with several implementations of attention mechanisms in the ablation study and find the classical scaled dot-product attention[16] performs consistently well on various tasks.

Decoding to the geometric space Finally, we decode the output sequence from the latent sequence through Physics-Cross-Attention according to the query pos_{out} of the output sampling locations and use another MLP to map the embeddings to the values of the output function

$$\begin{aligned} p^{(j)} &= \text{trunk-projector}(pos_{out}^{(j)}), && p^{(j)} \in \mathbb{R}^{D\times 1} \\ \boldsymbol{U} &= \text{Physics-Cross-Attention}(\boldsymbol{P}, \boldsymbol{H}, \boldsymbol{Z}^L) \\ val_{out}^{(j)} &= \text{MLP}(u^{(j)}), && u^{(j)} \in \mathbb{R}^{D\times 1} \end{aligned} \tag{3}$$

where $\boldsymbol{P} = [p^{(1)}, p^{(2)}, \ldots, p^{(N_{out})}]^T$ and $\boldsymbol{U} = [u^{(1)}, u^{(2)}, \ldots, u^{(N_{out})}]^T$. During the inferring phase, the trunk-projector can accept inputs from any position, enabling LNO to generalize to the unseen region where the positions are not presented in the training phase.

https://doi.org/10.52202/079017-1042

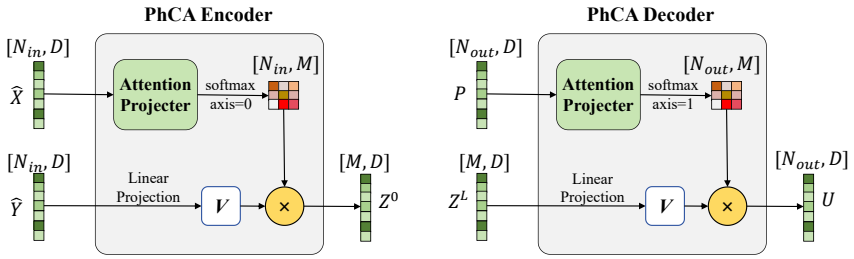


Figure 2: The working mechanism of Physics-Cross-Attention in encoder and decoder respectively.

3.3 Physics-Cross-Attention

The **Ph**ysics-**C**ross-**A**ttension (PhCA) is the core module in the aforementioned latent neural operator (LNO) model, acting as the encoder and decoder which transform the data representation back-and-forth between the real-world geometric space and the latent space. Motivated by the work [22] that learns the operator mapping across domains using the cross-attention, we use the same cross-attention to model the transformation to/from the latent space but set the latent space learnable rather than pre-defined, *e.g.,* triangular space or frequency domain [14].

In particular, the rows of the query $\boldsymbol{H}$ and key $\hat{\boldsymbol{X}}$ matrices in (2) represent the embedded positions of samples in the target and source space respectively. A row of the value matrix $\hat{\boldsymbol{Y}}$ is the embedding of both the position and physics quantity of a sample in the source space. We design the latent space to be learned from data. Thus, the positions $\boldsymbol{H}$ of samples in the latent space should be set as learnable parameters as well. Then, the cross-attention of the encoder in (2) is simplified as

$$\boldsymbol{Z}^0 = \mathrm{softmax}\Big(\tfrac{1}{\sqrt{D}}\boldsymbol{H}\boldsymbol{W}_q\boldsymbol{W}_k^T\hat{\boldsymbol{X}}^T\Big)\hat{\boldsymbol{Y}}\boldsymbol{W}_v = \mathrm{softmax}\big(\boldsymbol{W}_1\hat{\boldsymbol{X}}^T\big)\hat{\boldsymbol{Y}}\boldsymbol{W}_v, \tag{4}$$

where we merge the learnable matrices of query $\boldsymbol{H}$, query weight $\boldsymbol{W}_q$ and key weight $\boldsymbol{W}_k$ as a single matrix $\boldsymbol{W}_1$.

Correspondingly, the cross-attention in the decoder has a learnable key matrix that represents positions of samples in the latent space. Therefore, the cross-attention of the decoder in (3) is written as

$$\boldsymbol{U} = \mathrm{softmax}\Big(\tfrac{1}{\sqrt{D}}\boldsymbol{P}\boldsymbol{W}_q^{'}\boldsymbol{W}_k^{'T}\boldsymbol{H}^T\Big)\boldsymbol{Z}^L\boldsymbol{W}_v^{'} = \mathrm{softmax}(\boldsymbol{P}\boldsymbol{W}_2^T)\boldsymbol{Z}^L\boldsymbol{W}_v^{'}. \tag{5}$$

In addition, utilizing the relationship of the encoder and decoder as mutually inverse transformations, we set $\boldsymbol{W}_1 = \boldsymbol{W}_2$ to reduce the number of parameters, which also improves model performance experimentally. Practically, we generalize the linear projection $\boldsymbol{W}_1, \boldsymbol{W}_2$ with an MLP for further improvement, which is named attention-projector as shown in Figure 2.

In the special case where the input only contains sample positions without physics quantities, our PhCA module is similar to the transformation of physics-aware token in Transolver [23] (only differing in normalization). In general cases when the inputs include both positions and physics quantities, the two approaches are different. Specifically, in Transolver, the key $\hat{\boldsymbol{X}}$ in (4) and query $\boldsymbol{P}$ in (5) contain both positions and physics quantities information, whereas in PhCA, the key matrix $\hat{\boldsymbol{X}}$ in the encoder and the query matrix $\boldsymbol{P}$ in the decoder only depend on the positions of the input and output respectively. The physics quantities are used only to produce the value matrix $\boldsymbol{V}$ in attention mechanism. Such decoupling property of values and locations in query and key enables LNO to predict values at different positions from input positions.

4 Experiments

We first evaluate the accuracy for the forward problems on the six popular public benchmarks and for the inverse problem using Burger's equation. Then, we measure the efficiency in terms of the number of model parameters, the GPU memory usage and the training time. Next, we assess the generalization of our model concerning the number N_{in} of observation samples and the number N_{out} of prediction positions. Finally, we establish a series of ablation studies including choices of different attention mechanisms, strategy of sharing weights and sampling numbers in the latent space. Model is developed on the PyTorch and the PaddlePaddle deep learning framework. Additional experiments on scaling, sampling rate, and significance are presented in the appendix.

4.1 Accuracy for the forward problems

We conduct experiments for the forward problem on six benchmarks including Darcy and NS2d on regular grids (refer to section 5.2, 5.3 in [14] for details), and Airfoil, Elasticity, Plasticity and Pipe problems on irregular grids (refer to section 4.1–4.5 in [44] for details).

Table 1 shows a comprehensive comparison with previous works that reported their performance on the same benchmarks. Our LNO achieves the best accuracy on four benchmarks: Darcy, NS2d, Elasticity and Pipe. On the Airfoil and Plasticity benchmarks, our LNO achieves performance close to the SOTA with only half computational cost as discussed in Section 4.3. These results demonstrate the effectiveness of approximating the kernel integral operator in the latent space after transformation from the real-world geometric space through the PhCA module.

Table 1: Prediction error on the six forward problems. The best result is in bold. "$*$" means that the results of the method are reproduced by ourselves. "/" means that the method can not handle this benchmark. The column labeled D.C. indicates whether the observation positions and prediction positions are decoupled.

Model	D.C.	Relative L2($\times 10^{-2}$)					
		Darcy	NS2d	Airfoil	Elasticity	Plasticity	Pipe
FNO[14]	N	1.08	15.56	/	/	/	/
Geo-FNO[44]	N	1.08	15.56	1.38	2.29	0.74	0.67
F-FNO*[45]	N	0.75	11.51	0.60	1.85	0.27	0.68
U-FNO*[46]	N	1.28	17.15	1.19	2.13	0.41	0.62
LSM[22]	N	0.65	15.35	0.59	2.18	0.25	0.50
Galerkin[20]	N	0.84	14.01	1.18	2.40	1.20	0.98
OFormer[29]	Y	1.24	17.05	1.83	1.83	0.17	1.68
GNOT*[21]	Y	1.04	13.40	0.75	0.88	3.19	0.45
FactFormer[30]	N	1.09	12.14	0.71	/	3.12	0.60
ONO[31]	Y	0.76	11.95	0.61	1.18	0.48	0.52
Transolver*[23]	N	0.58	8.79	**0.47**	0.62	**0.12**	0.31
LNO(Ours)	Y	**0.49**	**8.45**	0.51	**0.52**	0.29	**0.26**

4.2 Accuracy for the inverse problem

To demonstrate the flexibility of LNO, we design an inverse problem. Given the partially observed solution $u(x)$, the aim is to recover the complete $u(x)$ in a larger domain. Specifically, we test on 1D Burgers' equation

$$\frac{\partial}{\partial t}u(x,t) = 0.01\frac{\partial^2}{\partial x^2}u(x,t) - u(x,t)\frac{\partial}{\partial x}u(x,t), \quad x \in [0,1],\ t \in [0,1]$$
$$u(x,0) \sim \text{GaussianProcess}(0, \exp\big(-\tfrac{2}{pl^2}\sin^2(\pi||x-x^{'}||\big)), \quad u(0,t) = u(1,t)$$

The ground-truth data is generated on a 128×128 grid with the periodic boundary condition. Initial conditions are sampled from the Gaussian process with the periodic length $p = 1$, scaling factor $l = 1$.

The objective of this inverse problem is to reconstruct the complete solution $u(x)$ in the whole spatiotemporal domain $(x,t) \in [0,1] \times [0,1]$ based on sparsely random-sampled or fixed-sampled observation in the sub-spatiotemporal domain $(x,t) \in [0,1] \times [0.25, 0.75]$, as illustrated in Figure 6 in the appendix.

Instead of the naive approach which directly learns the mapping from partially observed samples in the subdomain to the complete solution in the whole domain, we propose a two-stage strategy inspired by inpainting[47, 48] and outpainting[49, 50]. First, we train the model as a completer to interpolate the sparsely sampled points in the subdomain $[0,1] \times [0.25, 0.75]$ (left sub-figure in Figure 6) to predict all densely and regularly sampled points in the same subdomain (middle sub-figure in Figure 6). Then, we train the model as a propagator to extrapolate the results of the completer from the subdomain to the whole domain $[0,1] \times [0,1]$ (right sub-figure in Figure 6). Since the observation and the prediction samples are located in different positions, only the models with decoupling properties can be used as the completer and propagator.

https://doi.org/10.52202/079017-1042

We compare the performance of our LNO with DeepONet[13] and GNOT[21] in two stages respectively. The results in Table 2 and Table 3 indicate that i) in the first stage, LNO performs significantly better than DeepONet and GNOT at different random observation ratios; ii) in the second stage, LNO performs better when the complete solution in the subdomain approaches the ground truth, and the performance degrades as the reconstruction error of the complete solution in the subdomain increases. This is reasonable since we train the propagator based on the ground truth in the subdomain. When there is a significant discrepancy between the ground truth and the prediction of the completer, the model faces challenging distribution shifts. Nevertheless, when LNO is used both as the completer and propagator, it achieves the most accurate results for solving this inverse problem.

We also investigate the impact of different temporal and spatial sampling intervals on the accuracy of LNO in the fixed grid observation situation. See appendix B for details.

Table 2: Relative MAE of different completers in the subdomain in the 1st stage of the inverse problem with different settings of observation ratio.

Completer | Observation ratio	20%	10%	5%	1%	0.5%
DeepONet[13]	2.51%	2.59%	2.82%	3.25%	4.82%
GNOT[21]	1.12%	1.39%	1.62%	1.63%	2.56%
LNO(Ours)	**0.60%**	**0.74%**	**0.77%**	**1.18%**	**2.05%**

Table 3: The reconstruction error of different propagators in the 2nd stage of the inverse problem. Propagators are trained to reconstruct the solution in the whole domain based on the ground truth (G.T.) of the subdomain and are tested using the output of different completers. Relative MAE of $t = 0$ and $t = 1$ is reported.

Propagator | Completer	G.T.	LNO		GNOT		DeepONet	
		10%	1%	10%	1%	10%	1%
DeepONet[13]	7.34%	8.01%	9.38%	9.09%	10.80%	11.14%	13.87%
GNOT[21]	5.45%	6.50%	8.07%	**8.04%**	**9.91%**	**10.41%**	**13.45%**
LNO(Ours)	**3.73%**	**5.69%**	**7.72%**	9.03%	10.98%	13.11%	15.50%

Table 4: Comparison of efficiency between LNO and Transolver on the six forward problem benchmarks. Three metrics are measured: the number of parameters, the cost of GPU memory and the cost of training time per epoch.

Metric	Model	Darcy	NS2d	Airfoil	Elasticity	Plasticity	Pipe
Paras Count(M)	Transolver	1.91	5.33	1.91	1.91	1.91	1.91
	LNO	**0.76**	**5.08**	**1.36**	**1.42**	**1.36**	**1.36**
Memory(GB)	Transolver	17.11	17.17	4.49	1.48	18.41	5.94
	LNO	**5.75**	**7.58**	**2.47**	**1.39**	**7.16**	**2.89**
Time(s/epoch)	Transolver	88.68	107.62	19.49	5.66	83.43	25.56
	LNO	**38.98**	**57.83**	**9.35**	**5.33**	**41.62**	**14.13**

4.3 Efficiency

We demonstrate improvement in the efficiency of our proposed LNO compared to Transolver, which was considered the most efficient Transformer-based PDE solvers[23]. Results in Table 4 indicate that our method has fewer parameters, lower memory consumption, and faster training speed. Roughly speaking, LNO reduces the parameter count by an average of 30% and memory consumption by 50% compared to Transolver across all benchmarks, also with a 1.8x training acceleration.

The improvement of LNO stems from the idea of solving PDEs in the latent space. Given N input points in the real-world geometric space, if both Transolver and LNO consist of L Transformer blocks, the Transolver, which applies Physics-Attention in each block, has the time complexity

of $O(LMN + LM^2)$, where M is the number of slices. Our LNO, which applies Physics-Cross-Attention only in the encoder and decoder, has the time complexity of $O(MN + LM^2)$, where M is the sampling numbers in latent space.

4.4 Generalization

To validate the generalization of our model concerning the number of discretization points in the input and output, we train the model in a uniform low-resolution scenario and perform zero-shot inference in other higher-resolution scenarios. Specifically, we downsample the Darcy dataset with a resolution of 421×421 to 241×241, 211×211, 141×141, 85×85, 61×61 and 43×43. Models are trained on the 43×43 dataset and tested on the other datasets. The comparison results of our model and other methods shown in Table 5 show that our model exhibits great generalization capability regarding the number of sampling points, consistently outperforming other methods in accuracy on test sets with a different number of discretization points from the training set.

Table 5: Prediction errors on Darcy with different resolutions. Models are trained on 43×43 and tested on higher resolutions. Relative L2 errors are reported. The best results are in bold.

Model	61×61	85×85	141×141	211×211	241×241
FNO	0.1164	0.1797	0.2679	0.3160	0.3631
ONO	0.0204	0.0259	0.0315	0.0349	0.0386
GNOT	0.0256	0.0275	0.0294	0.0305	0.0317
Transolver	0.0239	0.0240	0.0258	0.0279	0.0290
LNO(Ours)	**0.0179**	**0.0192**	**0.0214**	**0.0229**	**0.0244**

4.5 Ablation Study

Necessity of decoupling Decoupling the values and locations as well as the observation positions and prediction positions not only guarantees flexibility but also improves accuracy. To study the effect of decoupling property, we replace the PhCA in LNO with Physics-Attention proposed in Transolver[23]. In this way, the query and keys are calculated from the same embedding of both pos_{in} and val_{in} of the input sequence. The pos_{out} which is not present in the input sequence can not generate queries and keys. So the observation positions and prediction positions must be kept the same. We compare the performance of LNO with PhCA or Physics-Attention on the forward problem benchmarks, and the results in the first and last rows of Table 6 verify the necessity of decoupling.

Expressiveness of quadratic attention We study the impact on accuracy using different attention mechanisms in the latent space. We replace the scaled dot-product attention in LNO with Galerkin-type attention used in Galerkin Transformer[20], efficient attention[18] used in GNOT[21], and Nystrom attention[19] used in ONO[31]. The comparison results in Table 6 show that quadratic attention, *i.e.,* the scaled dot-product attention[16], is more expressive than linear attention, which helps improve the accuracy. Many previous methods have employed different linear attentions[20, 21, 31] to reduce the computational costs at the expense of reducing the accuracy. Since the length of the latent sequence is much smaller than the inputs and outputs in real-world geometric space, applying quadratic attention in latent space does not incur significant overhead.

Sharing parameters We explore the performance of LNO when $W_1 \neq W_2$ in (4) and (5). In this way, we replace the linear projection W_1 and W_2 with two individual MLPs. Results of the comparison between LNO using shared MLPs and non-shared MLPs in the PhCA encoder and decoder are shown in Table 7. It can be found that LNO performs better on most benchmarks (except Plasticity) under the shared situation.

Sampling numbers in latent space We investigate the impact of hypothetical sampling position number in the latent space on LNO for solving both the forward and inverse problems. Results are shown in Figure 3. In the forward problems, as the sampling number M in the latent space increases, the performance of LNO gradually improves to a saturation level, followed by a gentle decline on the benchmarks of Airfoil, Elasticity, Plasticity, and Pipe. However, on benchmarks of Darcy and NS2d, the performance of LNO continues to improve till $M = 512$. This may be attributed to that

https://doi.org/10.52202/079017-1042

the former set of benchmarks containing only location information, while the latter set involves both location and physics quantity information, which requires more sampling positions to represent. In the inverse problem, the performance of the completer and propagator are similar to the trends in the forward problems.

Table 6: The ablation results using different attention implementations. P.A. stands for Physics-Attention. E.A. stands for Efficient Attention, N.A. stands for Nystrom Attention and G.A. stands for Galerkin-type Attention.

Model	Relative L2($\times 10^{-2}$)					
	Darcy	NS2d	Airfoil	Elasticity	Plasticity	Pipe
LNO with P.A.[23]	0.53	22.65	0.74	0.88	0.49	0.40
LNO with E.A.[18, 21]	0.63	20.18	0.59	0.54	0.44	0.36
LNO with N.A.[19, 31]	0.67	23.03	0.63	0.79	0.30	0.48
LNO with G.A.[20]	0.71	22.44	0.89	0.93	0.43	0.39
LNO	**0.49**	**8.45**	**0.51**	**0.52**	**0.29**	**0.26**

Table 7: The ablation results of strategy of sharing weights on six forward problem benchmarks. Relative L2 is recorded. Smaller values represent better performance. The better result is in bold.

Model	Relative L2($\times 10^{-2}$)					
	Darcy	NS2d	Airfoil	Elasticity	Plasticity	Pipe
LNO(non-shared)	0.54	10.67	0.53	1.09	**0.26**	0.36
LNO	**0.49**	**8.45**	**0.51**	**0.52**	0.29	**0.26**

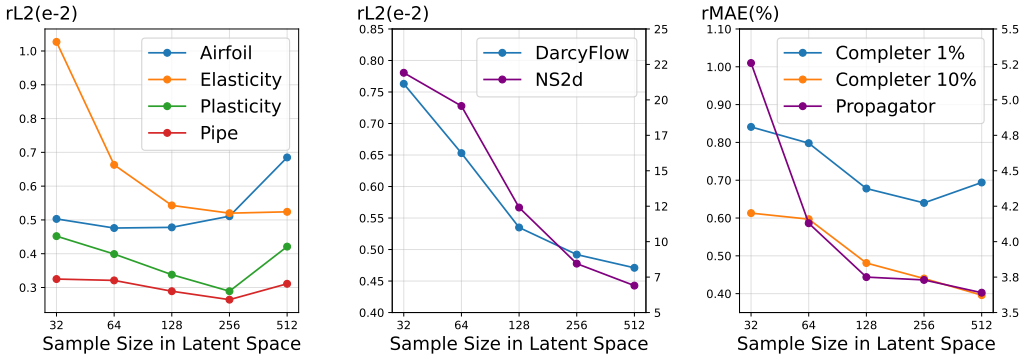


Figure 3: Sampling numbers in latent space v.s. Accuracy. (a) Forward problems on Airfoil, Elasticity, Plasticity and Pipe. (b) Forward problems on Darcy and NS2d. (c) Inverse Problem.

5 Conclusions

The paper explores the feasibility of characterizing PDEs in a learnable latent space. We present **Ph**ysics-**C**ross-**A**ttension (PhCA) for encoding inputs into the latent space and for decoding outputs. We build the Latent Neural Operator (LNO) based on PhCA and validate its flexibility, efficiency and accuracy for solving both forward and inverse problems. We leave the generalization ability and re-usability of PhCA across multiple types of PDEs as another open research topic.

Our work has also some limitations. For time-dependent equations, we decode the states at intermediate time steps into geometric space to compute the loss. Implementing additional training strategies may allow us to conduct all operations of the dynamic evolution process directly in the latent space, which potentially enhances efficiency. Also, in our work, no prior is imposed on the latent space. However, in the future, it is worth exploring whether physical priors, *e.g.,* symmetry, can help gain further precision improvement.

https://doi.org/10.52202/079017-1042

References

[1] Jaideep Pathak, Shashank Subramanian, Peter Harrington, Sanjeev Raja, Ashesh Chattopadhyay, Morteza Mardani, Thorsten Kurth, David Hall, Zongyi Li, Kamyar Azizzadenesheli, et al. FourCastNet: a global data-driven high-resolution weather model using adaptive Fourier neural operators. *arXiv preprint arXiv:2202.11214*, 2022.

[2] Zhenze Yang, Chi-Hua Yu, and Markus J Buehler. Deep learning model to predict complex stress and strain fields in hierarchical composites. *Science Advances*, 7(15):eabd7416, 2021.

[3] Yifan Mei, Yijie Zhang, Xueyu Zhu, and Rongxi Gou. Forward and inverse problems for eikonal equation based on DeepONet. *arXiv preprint arXiv:2306.05754*, 2023.

[4] Timothy Praditia, Matthias Karlbauer, Sebastian Otte, Sergey Oladyshkin, Martin V Butz, and Wolfgang Nowak. Learning groundwater contaminant diffusion-sorption processes with a finite volume neural network. *Water Resources Research*, 58(12):e2022WR033149, 2022.

[5] Junuthula Narasimha Reddy. An introduction to the finite element method. *New York*, 27:14, 1993.

[6] Zhongkai Hao, Songming Liu, Yichi Zhang, Chengyang Ying, Yao Feng, Hang Su, and Jun Zhu. Physics-informed machine learning: a survey on problems, methods and applications. *arXiv preprint arXiv:2211.08064*, 2022.

[7] Maziar Raissi, Paris Perdikaris, and George E Karniadakis. Physics-informed neural networks: a deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019.

[8] Matthias Karlbauer, Timothy Praditia, Sebastian Otte, Sergey Oladyshkin, Wolfgang Nowak, and Martin V Butz. Composing partial differential equations with physics-aware neural networks. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2022.

[9] Craig R Gin, Daniel E Shea, Steven L Brunton, and J Nathan Kutz. DeepGreen: deep learning of Green's functions for nonlinear boundary value problems. *Scientific Reports*, 11(1):21614, 2021.

[10] Chengping Rao, Pu Ren, Qi Wang, Oral Buyukozturk, Hao Sun, and Yang Liu. Encoding physics to learn reaction–diffusion processes. *Nature Machine Intelligence*, 5(7):765–779, 2023.

[11] Sifan Wang, Hanwen Wang, and Paris Perdikaris. Learning the solution operator of parametric partial differential equations with physics-informed DeepONets. *Science Advances*, 7(40):eabi8605, 2021.

[12] Tianping Chen and Hong Chen. Universal approximation to nonlinear operators by neural networks with arbitrary activation functions and its application to dynamical systems. *IEEE Transactions on Neural Networks*, 6(4):911–917, 1995.

[13] Lu Lu, Pengzhan Jin, and George Em Karniadakis. DeepONet: learning nonlinear operators for identifying differential equations based on the universal approximation theorem of operators. *arXiv preprint arXiv:1910.03193*, 2019.

[14] Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. *arXiv preprint arXiv:2010.08895*, 2020.

[15] Nikola Kovachki, Zongyi Li, Burigede Liu, Kamyar Azizzadenesheli, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Neural operator: learning maps between function spaces with applications to PDEs. *Journal of Machine Learning Research*, 24(89):1–97, 2023.

[16] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[17] Georgios Kissas, Jacob H Seidman, Leonardo Ferreira Guilhoto, Victor M Preciado, George J Pappas, and Paris Perdikaris. Learning operators with coupled attention. *Journal of Machine Learning Research*, 23(215):1–63, 2022.

[18] Zhuoran Shen, Mingyuan Zhang, Haiyu Zhao, Shuai Yi, and Hongsheng Li. Efficient attention: attention with linear complexities. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 2021.

 https://doi.org/10.52202/079017-1042

[19] Yunyang Xiong, Zhanpeng Zeng, Rudrasis Chakraborty, Mingxing Tan, Glenn Fung, Yin Li, and Vikas Singh. Nyströmformer: a Nyström-based algorithm for approximating self-attention. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2021.

[20] Shuhao Cao. Choose a transformer: Fourier or Galerkin. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[21] Zhongkai Hao, Zhengyi Wang, Hang Su, Chengyang Ying, Yinpeng Dong, Songming Liu, Ze Cheng, Jian Song, and Jun Zhu. GNOT: a general neural operator Transformer for operator learning. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2023.

[22] Haixu Wu, Tengge Hu, Huakun Luo, Jianmin Wang, and Mingsheng Long. Solving high-dimensional PDEs with latent spectral models. *arXiv preprint arXiv:2301.12664*, 2023.

[23] Haixu Wu, Huakun Luo, Haowen Wang, Jianmin Wang, and Mingsheng Long. Transolver: a fast Transformer solver for PDEs on general geometries. *arXiv preprint arXiv:2402.02366*, 2024.

[24] Benedikt Alkin, Andreas Fürst, Simon Schmid, Lukas Gruber, Markus Holzleitner, and Johannes Brandstetter. Universal physics Transformers. *arXiv preprint arXiv:2402.12365*, 2024.

[25] Valerii Iakovlev, Markus Heinonen, and Harri Lähdesmäki. Learning space-time continuous latent neural PDEs from partially observed states. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

[26] Tailin Wu, Takashi Maruyama, and Jure Leskovec. Learning to accelerate partial differential equations via latent global evolution. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[27] Somdatta Goswami, Katiana Kontolati, Michael D Shields, and George Em Karniadakis. Deep transfer operator learning for partial differential equations under conditional shift. *Nature Machine Intelligence*, 4(12):1155–1164, 2022.

[28] Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Neural operator: graph kernel network for partial differential equations. *arXiv preprint arXiv:2003.03485*, 2020.

[29] Zijie Li, Kazem Meidani, and Amir Barati Farimani. Transformer for partial differential equations' operator learning. *arXiv preprint arXiv:2205.13671*, 2022.

[30] Zijie Li, Dule Shu, and Amir Barati Farimani. Scalable Transformer for PDE surrogate modeling. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

[31] Zipeng Xiao, Zhongkai Hao, Bokai Lin, Zhijie Deng, and Hang Su. Improved operator learning by orthogonal attention. *arXiv preprint arXiv:2310.12487*, 2023.

[32] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2008.

[33] Ruchi Guo, Shuhao Cao, and Long Chen. Transformer meets boundary value inverse problems. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2022.

[34] Guanchao Liu, Lei Zhang, Qingzhen Wang, and Jianhua Xu. Data-driven seismic prestack velocity inversion via combining residual network with convolutional autoencoder. *Journal of Applied Geophysics*, 207:104846, 2022.

[35] Bin Liu, Yuxiao Ren, Hanchi Liu, Hui Xu, Zhengfang Wang, Anthony G Cohn, and Peng Jiang. GPRInvNet: deep learning-based ground-penetrating radar data inversion for tunnel linings. *IEEE Transactions on Geoscience and Remote Sensing*, 59(10):8305–8325, 2021.

[36] Zhongping Zhang, Yue Wu, Zheng Zhou, and Youzuo Lin. VelocityGAN: subsurface velocity image estimation using conditional adversarial networks. In *Proceedings of the IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2019.

[37] Juri Ranieri, Amina Chebira, Yue M Lu, and Martin Vetterli. Sampling and reconstructing diffusion fields with localized sources. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2011.

[38] Hassan Bararnia and Mehdi Esmaeilpour. On the application of physics informed neural networks (PINN) to solve boundary layer thermal-fluid problems. *International Communications in Heat and Mass Transfer*, 132(8):105890, 2022.

[39] Shengze Cai, Zhicheng Wang, Chryssostomos Chryssostomidis, and George Em Karniadakis. Heat transfer prediction with unknown thermal boundary conditions using physics-informed neural networks. In *Proceedings of the Fluids Engineering Division Summer Meeting (FEDSM)*, 2020.

[40] Hongping Wang, Yi Liu, and Shizhao Wang. Dense velocity reconstruction from particle image velocimetry/particle tracking velocimetry using a physics-informed neural network. *Physics of Fluids*, 34(1):017116, 2022.

[41] Roberto Molinaro, Yunan Yang, Björn Engquist, and Siddhartha Mishra. Neural inverse operators for solving PDE inverse problems. *arXiv preprint arXiv:2301.11167*, 2023.

[42] Coşku Can Horuz, Matthias Karlbauer, Timothy Praditia, Martin V Butz, Sergey Oladyshkin, Wolfgang Nowak, and Sebastian Otte. Infering boundary conditions in finite volume neural networks. In *Proceedings of the International Conference on Artificial Neural Networks (ICANN)*, 2022.

[43] Qingqing Zhao, David B Lindell, and Gordon Wetzstein. Learning to solve pde-constrained inverse problems with graph networks. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2022.

[44] Zongyi Li, Daniel Zhengyu Huang, Burigede Liu, and Anima Anandkumar. Fourier neural operator with learned deformations for PDEs on general geometries. *Journal of Machine Learning Research*, 24(388):1–26, 2023.

[45] Alasdair Tran, Alexander Mathews, Lexing Xie, and Cheng Soon Ong. Factorized Fourier neural operators. *arXiv preprint arXiv:2111.13802*, 2021.

[46] Gege Wen, Zongyi Li, Kamyar Azizzadenesheli, Anima Anandkumar, and Sally M Benson. U-FNO–an enhanced Fourier neural operator-based deep-learning model for multiphase flow. *Advances in Water Resources*, 163:104180, 2022.

[47] Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei A Efros. Context encoders: Feature learning by inpainting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[48] Jiahui Yu, Zhe Lin, Jimei Yang, Xiaohui Shen, Xin Lu, and Thomas S Huang. Generative image inpainting with contextual attention. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[49] Zongxin Yang, Jian Dong, Ping Liu, Yi Yang, and Shuicheng Yan. Very long natural scenery image prediction by outpainting. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019.

[50] Mark Sabini and Gili Rusak. Painting outside the box: Image outpainting with gans. *arXiv preprint arXiv:1808.08483*, 2018.

[51] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2018.

[52] Leslie N Smith and Nicholay Topin. Super-convergence: very fast training of neural networks using large learning rates. In *Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications*, 2019.

 https://doi.org/10.52202/079017-1042

A Scaling

We conduct scaling experiments to investigate the impact of model depth (number of stacked Transformer blocks) and model width (dimension of tokens in Transformer blocks) on the performance of LNO.

As shown in Figure 4, with model depth increasing, the performance of LNO initially improves and then declines, reaching its optimal performance at 4 or 8. This suggests that deeper LNO encounters optimization challenges in solving PDEs in the latent space. Simply stacking more Transformer blocks does not necessarily lead to better accuracy.

As shown in Figure 5, although the performance of LNO continues benefiting from increasing model width, it gradually exhibits a saturation trend when the dimension of tokens reaches as large as 256.

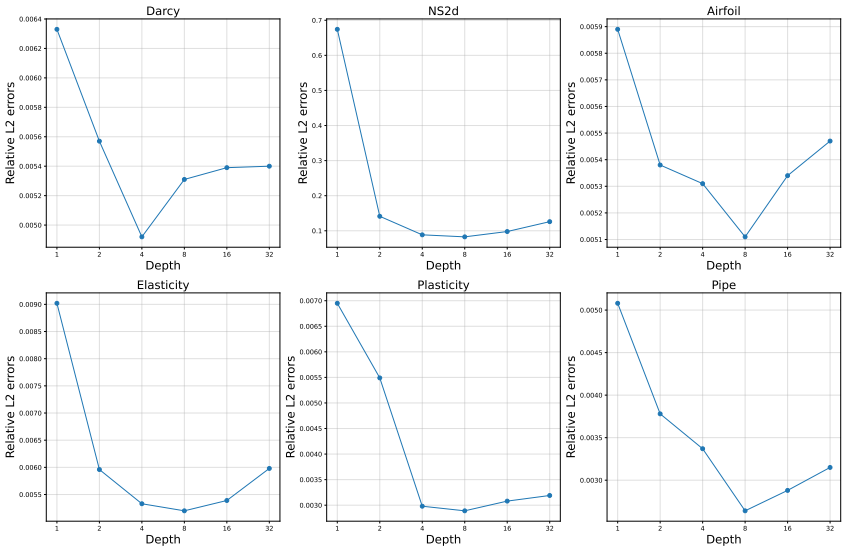


Figure 4: The influence of model depth on the performance of LNO across six forward problem benchmarks.

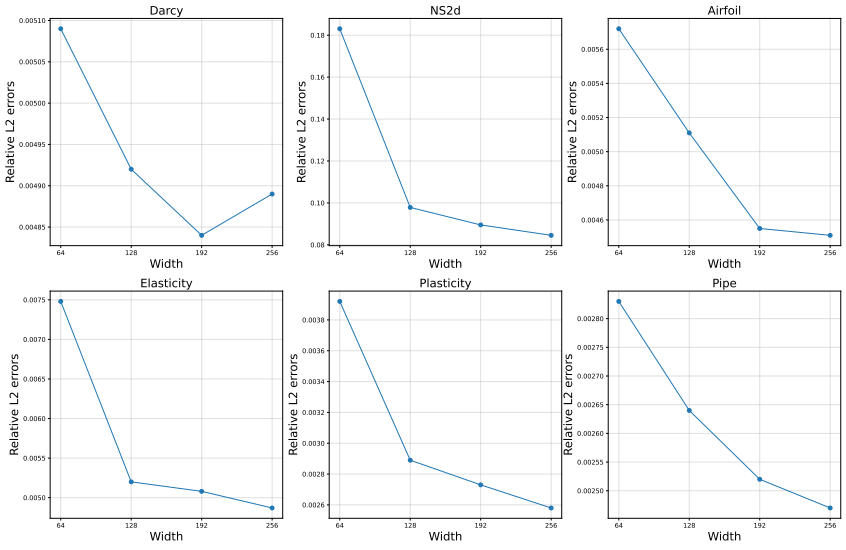


Figure 5: The influence of model width on the performance of LNO across six forward problem benchmarks.

https://doi.org/10.52202/079017-1042

B Observation Ratio and Locations for the Inverse PDE Task

We investigate the impact of different temporal and spatial sampling intervals on the accuracy of LNO when solving inverse problem in the fixed observation situation. As shown by Figure 8, it can be found that i) the inverse problem can still be solved accurately even when the temporal and spatial sampling intervals are both as large as 16 (approximately only 0.4% observation ratio); ii) the increase of the spatial sampling interval significantly compromises the solution accuracy compared to that of the temporal sampling interval; iii) spatial undersampling can hardly be compensated for through temporal oversampling.

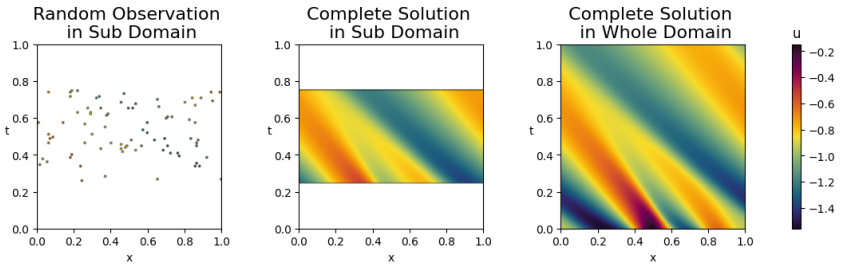


Figure 6: Visualization of Burgers' equation with different observation situations in different regions. We propose a two-stage strategy. First, we interpolate the solution in the subdomain. Then, we extrapolate from the subdomain to the whole domain.

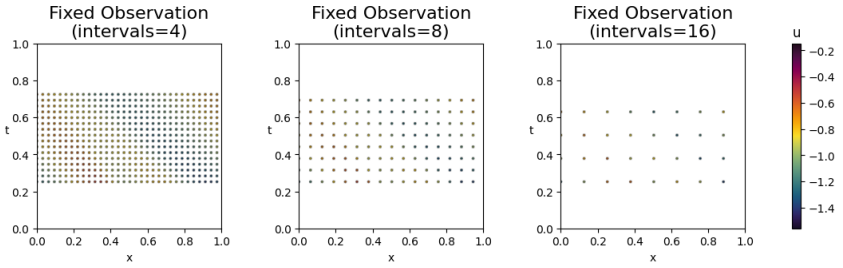
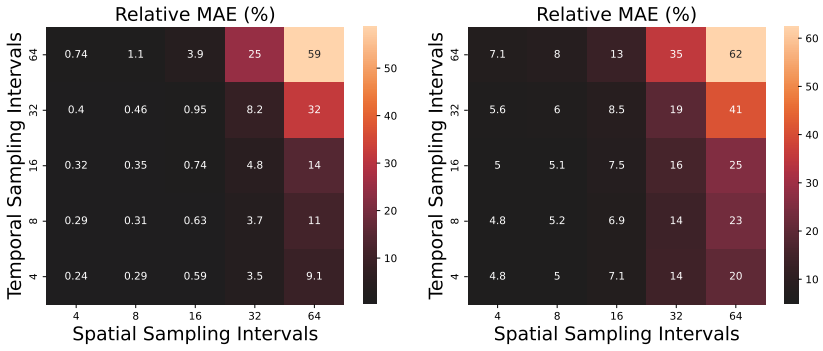


Figure 7: Visualization of Burgers' equation in the fixed observation situation with different temporal and spatial sampling intervals.



(a) (b)

Figure 8: Accuracy of solving the inverse problem at different temporal and spatial sampling intervals. (a) Accuracy of LNO as completer. (b) Accuracy of LNO as propagator based on the results of the corresponding completer.

https://doi.org/10.52202/079017-1042

C Significance

We conduct repeated experiments on both the forward and inverse problems, initializing the weights of our model and the reproduced models with different random seeds each time. We report the mean and standard deviation of these experiments to demonstrate the significance of our experimental results, as shown in Figure 8 and Figure 9.

Table 8: The mean and standard deviations (computed based on 5 independent trials) of different models in solving the forward problem.

Model	Relative L2($\times 10^{-2}$)					
	Darcy	NS2d	Airfoil	Elasticity	Plasticity	Pipe
F-FNO[45]	0.75($\pm$0.01)	11.51($\pm$0.25)	0.60($\pm$0.03)	1.85($\pm$0.03)	0.27($\pm$0.02)	0.68($\pm$0.02)
U-FNO[46]	1.28($\pm$0.02)	17.15($\pm$0.29)	1.19($\pm$0.05)	2.13($\pm$0.03)	0.41($\pm$0.03)	0.62($\pm$0.02)
GNOT[21]	1.04($\pm$0.02)	13.40($\pm$0.28)	0.75($\pm$0.02)	0.88($\pm$0.03)	3.19($\pm$0.07)	0.45($\pm$0.02)
Transolver[23]	0.58($\pm$0.01)	8.79($\pm$0.17)	0.47($\pm$0.02)	0.62($\pm$0.02)	0.12($\pm$0.02)	0.31($\pm$0.01)
LNO(Ours)	0.49($\pm$0.01)	8.45($\pm$0.22)	0.51($\pm$0.05)	0.52($\pm$0.03)	0.29($\pm$0.03)	0.26($\pm$0.03)

Table 9: The mean and standard deviations (computed based on 3 independent trials) of different models in solving the inverse problem. Relative MAEs are expressed as percentages. The second to the second-to-last columns correspond to the situation where models act as completers under different observation rates, while the last column corresponds to the situation where models act as propagators.

Model	20%	10%	5%	1%	0.5%	propagator
DeepONet[13]	2.37($\pm$0.18)	2.52($\pm$0.04)	2.77($\pm$0.10)	3.18($\pm$0.03)	4.70($\pm$0.31)	7.42($\pm$0.37)
GNOT[21]	1.23($\pm$0.07)	1.33($\pm$0.06)	1.63($\pm$0.04)	1.75($\pm$0.02)	2.53($\pm$0.08)	5.43($\pm$0.10)
LNO(Ours)	0.60($\pm$0.03)	0.73($\pm$0.11)	0.78($\pm$0.04)	1.26($\pm$0.09)	1.97($\pm$0.04)	3.46($\pm$0.21)

D Full Implementation Details

The full implementation details of LNO in both forward and inverse problems are shown in Table 10. All our experiments are conducted on single or two NVIDIA RTX 3090 GPUs.

For the forward problems, we train the models using relative L2 error for 500 epochs. AdamW optimizer[51] and OneCycleLr scheduler[52] are applied with initial learning rate at 10^{-3}. We reproduce the results of Transolver[23] following implementations in appendix B.3 in [23] and also excerpt the results of other methods from Table 2 in [23]. The data splits are the same as in Appendix B.1 in [23].

For the inverse problem, we train the models using mean squared error and StepLr scheduler. For DeepONet[13], we use 4 Transformer blocks with 128-dimensional tokens as the branch net and 3 fully-connected layers as the trunk net. For GNOT[21], we set the number of Transformer blocks as 4, the dimension of tokens as 96, and the number of experts as 1. We use 4096 samples for training, 128 samples for validating and 128 samples for testing.

Table 10: The hyperparameters and training configuration of LNO in solving the forward and inverse problems in Section 4.1 and 4.2

Benchmark	Layers	Dim	Size	Heads	Batch	Epoch	Loss	Optimizer	Scheduler
Darcy	4	128	256	8	4	500	rL2	AdamW[51]	OneCycleLR[52]
NS2d	8	256							
Airfoil	8	128							
Elasticity	8	128							
Plasticity	8	128							
Pipe	8	128							
Completer	4	96	256	8	4	500	MSE	AdamW	StepLR
Propagator									

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We have clearly claimed the main contributions and scope of our work in the abstract and introduction, which can be reflected by the demonstration of methods in Section 3.2, 3.3 and experimental results in Section 4.1, 4.2, 4.3, 4.4, 4.5.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We have discussed the limitations of our proposed LNO and potential approaches to overcome them in Section 5.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

https://doi.org/10.52202/079017-1042

Answer: [NA]

Justification: Our work does not include theoretical results.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We provide detailed descriptions of our methods in Section 3.2 and Section 3.3, and full implementation details regarding the experiments in Appendix D. Additionally, we also provide our GitHub repository which contains the source code and running instructions.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: We have provided our source code along with detailed instructions on the runtime environment, data access, data processing, and experiment conducting in our GitHub repository.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [Yes]

Justification: We have provided the full implementation details in Appendix D including data splits, hyperparameters, optimizer, scheduler, epoch, etc. Also, the ablation studies in Section 4.5 and scaling experiments in Appendix A have shown the reasonableness of our choice for the hyperparameters.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: We have reported the experiment statistical significance in Appendix C, as shown in Table 8 and Table 9.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

https://doi.org/10.52202/079017-1042

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We have provided a comprehensive summary of computational resources required for all the experiments in Appendix D, along with detailed records of memory, execution time, etc., in Section 4.3.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We have reviewed the NeurIPS Code of Ethics carefully and made sure our research conform with it in every respect.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [NA]

    Justification: Our work is foundational research about solving forward and inverse problems of PDEs, which is not tied to particular applications or deployments. And there is no direct path to any negative applications.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: Our work primarily focuses on solving the forward and inverse problems of PDEs, which does not pose such risks of misuse.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We have utilized publicly available datasets provided in [14] and [44], as mentioned in Section 4.1. The URLs of the assets are also provided in our GitHub repository.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.

https://doi.org/10.52202/079017-1042

- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: We have provided the source code of our proposed LNO in our GitHub repository. Detailed documentation is also included to explain how to use our code.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: Our work does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: Our work does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.

- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

https://doi.org/10.52202/079017-1042