Electronic Research Archive

https://www.aimspress.com/journal/era

ERA, 35(1): 1–44.
DOI: 10.3934/era.2027001
Received: 31 August 2026
Revised: 13 October 2026
Accepted: 30 October 2026
Published: 27 November 2026

Research article

Data-driven optical solitons and parameter discovery of the nonlinear Schrödinger equation with Kerr and non-Kerr nonlinearities

Zekang Wu[1], **Manwai Yuen**[2] **and Lijun Zhang**[3,4,*]

[1] College of Mathematics and Systems Science, Shandong University of Science and Technology, Qingdao 266590, China
[2] Department of Mathematics and Information Technology, The Education University of Hong Kong, Hong Kong, China
[3] School of Science, Zhejiang University of Science and Technology, Hangzhou 310023, China
[4] Material Science Innovation and Modelling (MaSIM) Research Focus Area, North-West University, Mafikeng Campus, Mmabatho 2735, South Africa

* **Correspondence:** Email: li-jun0608@163.com.

Abstract: The nonlinear Schrödinger equation (NLSE) with Kerr and non-Kerr nonlinearities is crucial for modeling optical soliton propagation in communications systems. Recently, deep learning has emerged as a powerful and versatile tool across scientific and engineering disciplines, particularly for addressing nonlinear partial differential equations (NPDEs). In this paper, we employed two deep neural network frameworks to investigate the NLSE with Kerr, power, parabolic, and logarithmic nonlinearities. Specifically, physics-informed neural networks (PINNs) were applied for solution reconstruction and parameter identification, while the deep operator network (DeepONet) was implemented to approximate the solution operator for fast inference. In forward problems, the relative $\mathbb{L}^2$-norm errors of PINNs range from $O(10^{-4})$ to $O(10^{-2})$. In contrast, DeepONet yielded errors between $O(10^{-3})$ and $O(10^{-2})$ over 2000 test samples. We further examined multiple factors affecting model performance, including network architecture, loss weighting, sensor quantity, dataset size, optimizer selection, batch size, and parameter perturbation. For PINNs, a residual-guided hybrid optimization strategy dynamically adjusts Adam iterations and was further tested on several NPDEs, while an adaptive learning rate schedule was deployed for DeepONet. For inverse problems, we identified key equation parameters and analyzed the effects of noise intensity and spatial content, parameter selection, observation time instants, and observation points on the inversion accuracy. These numerical results demonstrate the capability of deep learning methods in simulating optical soliton dynamics and discovering unknown parameters, while also offering insights for model extension and algorithm improvement.

Keywords: physics-informed neural networks; deep operator network; nonlinear Schrödinger equations; Kerr and non-Kerr law nonlinearities; data-driven optical solitons; parameter discovery

1. Introduction

Solitons are characterized by their ability to maintain localized waveforms even after interacting with other solitons. Consequently, the study of their stability and long-time asymptotic behavior has attracted considerable attention and led to remarkable progress [1, 2]. In addition, their applications in fields such as optics and telecommunications have garnered significant interest [3, 4]. The renowned nonlinear Schrödinger equation (NLSE) serves as a fundamental model in Kerr media and has been successfully applied to a wide range of systems, including quantum mechanics, Bose-Einstein condensates, shallow water wave propagation, pulse dynamics, and optical fibers. However, its application to optical systems faces two key limitations [5]. On the one hand, optical loss is unavoidable because any optical transmitting medium possesses a finite attenuation coefficient. On the other hand, Kerr nonlinearity becomes inadequate for describing soliton dynamics in ultra-high bit rate transmissions due to its non-saturable characteristics. To model propagation in highly nonlinear materials at such rates, it is crucial to incorporate higher-order nonlinearities, motivating the integration of non-Kerr nonlinearity into the NLSE framework [6].

The dimensionless form of the NLSE, as presented in [5], is given by

$$i\psi_t + \frac{1}{2}\psi_{xx} + \mathcal{F}(|\psi|^2)\psi = 0, \tag{1.1}$$

where $\mathcal{F}(\cdot)$ is a real-valued function of optical intensity $|\psi|^2$. In the case of Kerr nonlinearity, where $\mathcal{F}(|\psi|^2) = |\psi|^2$, Eq (1.1) reduces to the standard integrable NLSE, which not only admits a rich variety of localized wave solutions such as solitons [7], breathers [8], and rogue waves [9], but also exhibits elegant mathematical structures, including symmetries [10], conservation laws [11], and Lax pairs [12]. Over the past century, numerous powerful analytical techniques have been developed to explore exact solutions and integrability properties of NLSEs [13, 14]. Nevertheless, these methods are mathematically intricate and are applicable only to a limited class of integrable nonlinear partial differential equations (NPDEs). In parallel, model extensions, applications, and numerical simulations of NLSEs have attracted growing research interest in recent years [15–17].

With the revival of neural networks, deep learning methods have achieved remarkable success in the field of scientific computing, particularly in addressing both forward and inverse problems associated with NPDEs (see Ref. [18] and the references therein). In contrast to conventional numerical methods, deep learning provides a mesh-free framework capable of tackling multivariate and high-dimensional problems in complex geometrical domains. Currently, the prevailing methodologies for solving NPDEs via deep neural networks (DNNs) can be categorized into two paradigms: function approximation and operator approximation. The most representative method in the first category is physics-informed neural networks (PINNs) [19], which integrate PDE residuals and conservation laws directly into the loss function, providing an efficient framework for complex systems that inherently satisfy the underlying physics. Building upon this foundational work, various enhanced PINN methods have been proposed, leveraging domain decomposition [20], gradient-based optimization (e.g., gradient-enhanced training [21] and gradient-balanced strategy [22]), sampling strategy [23], adaptive activation functions [24], and other techniques [25, 26] to improve prediction accuracy and tackle complex models. Methods in the second category utilize DNNs to learn nonlinear mappings between function spaces. A landmark contribution in this field is the deep operator network (DeepONet) [27], which is

theoretically based on the universal approximation theorem for operators and demonstrates the capability to accurately approximate any continuous nonlinear operator [28]. In contrast to PINNs, a key advantage of DeepONet is that once the DNNs are well-trained, it enables instantaneous solution prediction for new input functions, thereby providing a powerful tool for real-time inference and scenarios requiring high-frequency model evaluations. Several extensions of DeepONet have been proposed in recent studies, such as MIONet, POD-DeepONet, PI-DeepONet, etc. (see Ref. [29] and the references therein). Among them, PI-DeepONet can be regarded as a model that greatly enhances and generalizes the capabilities of PINNs [30, 31], which reduces the reliance on training data but significantly increases the computational time due to derivative calculations based on automatic differentiation.

Currently, the majority of deep learning approaches are dedicated to canonical PDEs defined over the real domain. In fact, addressing the problem in the complex domain involves decomposing the PDE as well as its initial-boundary conditions into real and imaginary components, which are then solved in a manner analogous to handling coupled equations. Remarkably, the application of PINNs has demonstrated significant potential in addressing NPDEs in the complex domain. Specifically, PINN-based techniques have been developed to investigate the dynamics of the NLSE and derivative NLSE with respect to the soliton, breather, and rogue wave solutions [32, 33]. The forward and inverse problems associated with fractional, logarithmic and saturable NLSEs, incorporating the $\mathcal{PT}$-symmetric potentials, were successfully investigated in [34–36]. More recently, the Noether-constrained PINNs [37], augmented with conservation laws and a residual-adaptive sampling strategy, have been further developed to reconstruct soliton solutions of the NLSE, exhibiting excellent generalization capability. Furthermore, PINNs have been extended to study high-order and high-dimensional NLSEs in optical fibers [38, 39], demonstrating their excellent capabilities in exploring sophisticated physical phenomena. In contrast, operator learning for complex PDEs remains relatively limited, with existing work focusing largely on special cases such as the NLSE (1.1) with $\mathcal{F}(\cdot) = 0$ and certain optical-fiber models [40–42]. Additionally, a systematic investigation of Kerr and multiple non-Kerr NLSEs using both a PINN and DeepONet, together with analyses of optimization efficiency and inversion robustness, has not yet been carried out. The contributions of this paper are summarized as follows:

- A unified investigation of the NLSE with Kerr, power, parabolic, and logarithmic nonlinearities is provided. PINNs are utilized for both forward soliton reconstruction and inverse parameter discovery, whereas data-driven DeepONet is implemented to approximate the operator mapping initial conditions to solution spaces.
- A residual-guided hybrid optimization scheme for PINNs is developed, which dynamically adjusts Adam iterations based on PDE residuals. Several critical hyperparameters for DeepONet are systematically examined.
- The sensitivity of the identified coefficients to noise intensity and spatial characteristics (e.g., high-frequency oscillations and low-frequency baseline drifts) is analyzed.

The remainder of this paper is organized as follows. Section 2 introduces the frameworks of PINNs for data-driven optical soliton reconstruction and parameter discovery, together with the DeepONet for approximating the solution operator. In Section 3, the forward problems of the NLSE with Kerr and non-Kerr nonlinearities are investigated by analyzing the real part, imaginary part, and modulus of the learned solutions. Various factors that may affect the performance of PINNs and DeepONet are examined. In Section 4, we address the inverse parameter-identification task via PINNs. Ultimately, we provide a comprehensive summary and discussion in Section 5.

2. Methods

In this section, we begin by outlining the methodology of PINNs for reconstructing optical soliton solutions and discovering unknown parameters for Eq (1.1). Following that, we present a detailed exposition of the data-driven DeepONet method, which approximates the operator mapping initial conditions to solution spaces.

2.1. The scheme of PINNs

In contrast to traditional numerical methods, which rely on mesh discretization, require special treatment of complex-valued boundary conditions, and necessitate careful balancing of error associated with dispersion and nonlinearity, PINNs are mesh-free and provide a unified framework for solving both forward and inverse problems. Solving NPDEs in the complex domain via PINNs involves preprocessing of the governing equations along with their initial and boundary conditions. The complex-valued function $\psi(x,t)$ defined in Eq (1.1) can be decomposed into its real and imaginary parts as $\psi = u + iv$, where $u(x,t)$ and $v(x,t)$ are real-valued functions. Alternatively, one may adopt the polar representation $\psi = Re^{i\theta}$. However, the polar representation is prone to numerical explosions and gradient anomalies in the loss function as $R(x,t)$ tends to zero. Substituting the former Cartesian decomposition into Eq (1.1) yields

$$i\left[u_t + \frac{1}{2}v_{xx} + \mathcal{F}(u^2+v^2)v\right] - \left[v_t - \frac{1}{2}u_{xx} - \mathcal{F}(u^2+v^2)u\right] = 0, \tag{2.1}$$

which is equivalent to

$$\begin{cases} v_t - \frac{1}{2}u_{xx} - \mathcal{F}(u^2+v^2)u = 0, \\ u_t + \frac{1}{2}v_{xx} + \mathcal{F}(u^2+v^2)v = 0. \end{cases} \tag{2.2}$$

Incorporating the initial conditions and Dirichlet boundary conditions leads to the following system for the real and imaginary components:

$$\begin{cases} v_t - \frac{1}{2}u_{xx} - \mathcal{F}(u^2+v^2)u = 0, \quad x \in [x_0, x_1], t \in [t_0, t_1], \\ u_t + \frac{1}{2}v_{xx} + \mathcal{F}(u^2+v^2)v = 0, \quad x \in [x_0, x_1], t \in [t_0, t_1], \\ u(x,t_0) = \mathcal{I}_u(x,t_0), \quad x \in [x_0, x_1], \\ v(x,t_0) = \mathcal{I}_v(x,t_0), \quad x \in [x_0, x_1], \\ u(x_0,t) = \mathcal{B}_u(x_0,t), u(x_1,t) = \mathcal{B}_u(x_1,t), \quad t \in [t_0, t_1], \\ v(x_0,t) = \mathcal{B}_v(x_0,t), v(x_1,t) = \mathcal{B}_v(x_1,t), \quad t \in [t_0, t_1]. \end{cases} \tag{2.3}$$

Here, $\mathcal{I}_u(x,t_0)$ and $\mathcal{I}_v(x,t_0)$ denote the initial conditions for $u(x,t)$ and $v(x,t)$, respectively, while $\mathcal{B}_u(x,t)$ and $\mathcal{B}_v(x,t)$ represent the boundary conditions. Next, we propose a PINNs-based framework to tackle forward and inverse problems associated with Eq (2.3). This is accomplished by first constructing a neural network representation of the model and then designing the loss function that enforces the governing equations and initial-boundary conditions.

2.1.1. Exploring data-driven optical soliton solutions through PINNs

In this study, we employ a feedforward neural network (FNN) architecture with N fully connected hidden layers, each containing n neurons. The explicit affine mapping between successive layer pairs is

defined as

$$\begin{aligned} \text{input layer:} \quad & \mathbf{x}_0 = \{x; t\}, \\ \text{hidden layers:} \quad & \mathbf{x}_i = \sigma(\mathbf{W}_i \mathbf{x}_{i-1} + \mathbf{b}_i), i = 1, 2, ..., N-1, \\ \text{output layer:} \quad & \mathbf{x}_N = \mathbf{W}_N \mathbf{x}_{N-1} + \mathbf{b}_N, \end{aligned} \tag{2.4}$$

where $\mathbf{x}_0 \in \mathbb{R}^2$, $\mathbf{x}_i \in \mathbb{R}^n$, $\mathbf{x}_N \in \mathbb{R}^2$, and σ is an activation function. The weight matrix and bias vector of the i-th layer are represented as $\mathbf{W}_i$ and $\mathbf{b}_i$, respectively. Specifically, $\mathbf{W}_1 \in \mathbb{R}^{n\times 2}$, $\mathbf{W}_i \in \mathbb{R}^{n\times n}(i = 2, ..., N-1)$, $\mathbf{W}_N \in \mathbb{R}^{2\times n}$, $\mathbf{b}_i \in \mathbb{R}^n(i = 1, 2, ..., N-1)$, and $\mathbf{b}_N \in \mathbb{R}^2$. By leveraging automatic differentiation, the neural network will continuously update its parameters $\boldsymbol{\theta} = \{\mathbf{W}_i, \mathbf{b}_i\}_{i=1}^N$ following each training iteration. A schematic representation of PINNs designed to solve optical soliton solutions of the NLSE in Kerr and non-Kerr law media is presented in Figure 1.

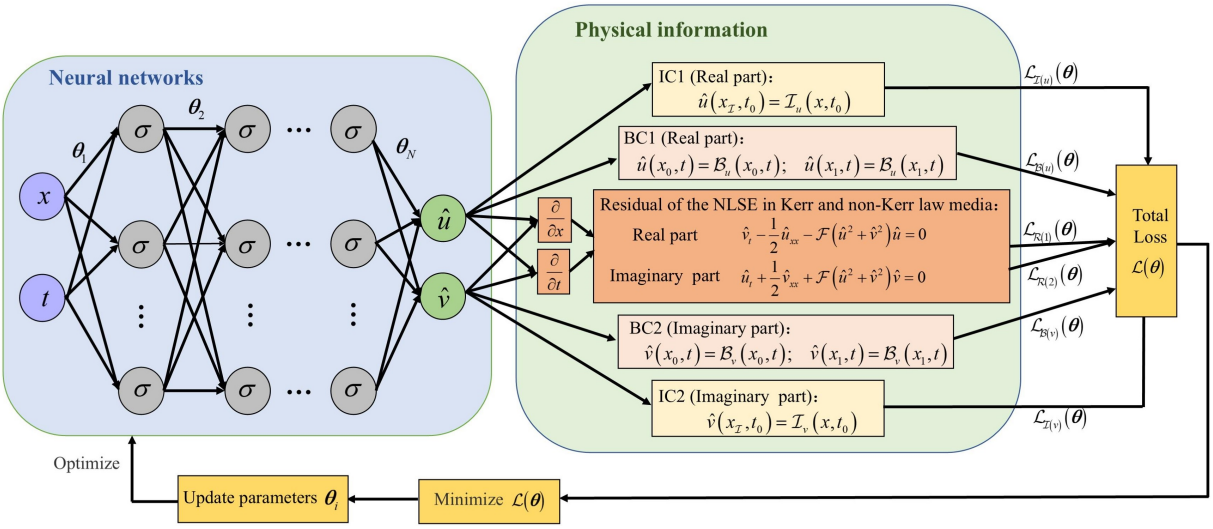


Figure 1. The PINNs framework for exploring data-driven optical soliton solutions of the NLSE in both Kerr and non-Kerr law media.

The loss function is subsequently introduced to quantify the discrepancy between the predictions and target values. This discrepancy arises from the network fitting the PDE residuals and the initial-boundary conditions. Specifically, the residuals of Eq (2.1) are defined as follows:

$$\begin{cases} \mathcal{R}_1(x,t) := \hat{v}_t - \frac{1}{2}\hat{u}_{xx} - \mathcal{F}(\hat{u}^2 + \hat{v}^2)\hat{u}, \\ \mathcal{R}_2(x,t) := \hat{u}_t + \frac{1}{2}\hat{v}_{xx} + \mathcal{F}(\hat{u}^2 + \hat{v}^2)\hat{v}. \end{cases} \tag{2.5}$$

Therefore, the total loss is given by

$$\mathcal{L}(\boldsymbol{\theta}) = w_{\mathcal{R}}\left[\mathcal{L}_{\mathcal{R}(1)}(\boldsymbol{\theta}) + \mathcal{L}_{\mathcal{R}(2)}(\boldsymbol{\theta})\right] + w_{\mathcal{I}}\left[\mathcal{L}_{\mathcal{I}(u)}(\boldsymbol{\theta}) + \mathcal{L}_{\mathcal{I}(v)}(\boldsymbol{\theta})\right] + w_{\mathcal{B}}\left[\mathcal{L}_{\mathcal{B}(u)}(\boldsymbol{\theta}) + \mathcal{L}_{\mathcal{B}(v)}(\boldsymbol{\theta})\right], \tag{2.6}$$

with

$$\begin{cases} \mathcal{L}_{\mathcal{R}(1)}(\boldsymbol{\theta}) = \frac{1}{N_{\mathcal{R}}} \sum_{j=1}^{N_{\mathcal{R}}} [\mathcal{R}_1(x_{\mathcal{R}}^j, t_{\mathcal{R}}^j)]^2, \quad \mathcal{L}_{\mathcal{R}(2)}(\boldsymbol{\theta}) = \frac{1}{N_{\mathcal{R}}} \sum_{j=1}^{N_{\mathcal{R}}} [\mathcal{R}_2(x_{\mathcal{R}}^j, t_{\mathcal{R}}^j)]^2, \\ \mathcal{L}_{\mathcal{I}(u)}(\boldsymbol{\theta}) = \frac{1}{N_{\mathcal{I}}} \sum_{j=1}^{N_{\mathcal{I}}} [\hat{u}(x_{\mathcal{I}}^j, t_0) - \mathcal{I}_u(x_{\mathcal{I}}^j, t_0)]^2, \quad \mathcal{L}_{\mathcal{I}(v)}(\boldsymbol{\theta}) = \frac{1}{N_{\mathcal{I}}} \sum_{j=1}^{N_{\mathcal{I}}} [\hat{v}(x_{\mathcal{I}}^j, t_0) - \mathcal{I}_v(x_{\mathcal{I}}^j, t_0)]^2, \\ \mathcal{L}_{\mathcal{B}(u)}(\boldsymbol{\theta}) = \frac{1}{N_{\mathcal{B}}} \sum_{j=1}^{N_{\mathcal{B}}} \left([\hat{u}(x_0, t_{\mathcal{B}}^j) - \mathcal{B}_u(x_0, t_{\mathcal{B}}^j)]^2 + [\hat{u}(x_1, t_{\mathcal{B}}^j) - \mathcal{B}_u(x_1, t_{\mathcal{B}}^j)]^2 \right), \\ \mathcal{L}_{\mathcal{B}(v)}(\boldsymbol{\theta}) = \frac{1}{N_{\mathcal{B}}} \sum_{j=1}^{N_{\mathcal{B}}} \left([\hat{v}(x_0, t_{\mathcal{B}}^j) - \mathcal{B}_v(x_0, t_{\mathcal{B}}^j)]^2 + [\hat{v}(x_1, t_{\mathcal{B}}^j) - \mathcal{B}_v(x_1, t_{\mathcal{B}}^j)]^2 \right), \end{cases} \tag{2.7}$$

where $\mathcal{L}_{\mathcal{R}(1)}(\boldsymbol{\theta})$ and $\mathcal{L}_{\mathcal{R}(2)}(\boldsymbol{\theta})$ represent the loss associated with residuals as defined in Eq (2.5). The terms $\mathcal{L}_{\mathcal{I}(u)}(\boldsymbol{\theta})$ and $\mathcal{L}_{\mathcal{I}(v)}(\boldsymbol{\theta})$ denote the loss related to initial conditions, while $\mathcal{L}_{\mathcal{B}(u)}(\boldsymbol{\theta})$ and $\mathcal{L}_{\mathcal{B}(v)}(\boldsymbol{\theta})$ represent the loss pertaining to boundary conditions. In addition, $N_{\mathcal{R}}$, $N_{\mathcal{I}}$, and $N_{\mathcal{B}}$ are the numbers of sampling points for the PDE residual, initial conditions, and boundary conditions, respectively. The symbols $\hat{u}$ and $\hat{v}$ represent the predicted solutions derived from the neural network's output. Furthermore, $w_{\mathcal{R}}$, $w_{\mathcal{I}}$, and $w_{\mathcal{B}}$ denote the weighting coefficients applied to each component of the loss function.

Subsequently, we select suitable optimizers to minimize the total loss (2.6) by updating the network parameters $\boldsymbol{\theta} = \{\mathbf{W}_j, \mathbf{b}_j\}_{j=1}^N$ within a finite number of iterations. Widely adopted optimizers for minimizing the loss function are adaptive moment estimation (Adam) and the limited-memory Broyden–Fletcher–Goldfarb–Shanno (L-BFGS). Adam is a first-order stochastic method that combines momentum with per-parameter adaptive learning rates, making it efficient for large-scale and non-convex problems like deep neural network training [43]. L-BFGS is a quasi-Newton method that leverages second-order information by iteratively building an approximation of the inverse Hessian using limited memory [44]. It converges rapidly in near-quadratic regions and is often used for convex optimization. The idea of combining Adam and L-BFGS was proposed to compensate for the shortcomings of a single optimizer, whose effectiveness has been validated in prior studies [36, 45]. The primary steps of the aforementioned PINNs method combined with mixed optimization in solving data-driven solutions of Eq (1.1) with Kerr and non-Kerr law media are concisely summarized in Table 1.

2.1.2. Discovering the parameters through PINNs

Drawing upon the information derived from the solution of the equations, the network parameters $\boldsymbol{\theta} = \{\mathbf{W}_i, \mathbf{b}_i\}_{i=1}^N$ are optimized to derive the coefficients of the equation. Based on the methodology employed to address the forward problem, we subsequently introduce the procedure for addressing the inverse problem of Eq (1.1) using PINNs. Equation (1.1), along with several undetermined parameters $\boldsymbol{\lambda} = (\lambda_1, \lambda_2, \mu, \nu)$, is presented as follows:

$$i\psi_t + \lambda_1 \psi_{xx} + \lambda_2 \mathcal{F}(|\psi|^2; \mu, \nu)\psi = 0. \tag{2.8}$$

Apart from the initial-boundary conditions specified in Eq (2.3), we possess information regarding $u_{\mathcal{D}}(x, t')$ and $v_{\mathcal{D}}(x, t')$ at designated locations within the spatial-temporal domain $\Omega \subset [x_0, x_1] \times [t_0, t_1]$. Therefore, the trainable parameters $\boldsymbol{\lambda} = (\lambda_1, \lambda_2, \mu, \nu)$ of Eq (2.8) can be accurately determined on the basis of these observations. Specifically, the loss of observations at the points $\left\{x_{\mathcal{D}}^j, t'\right\}_{j=1}^{N_{\mathcal{D}}}$ for $u_{\mathcal{D}}$ and $v_{\mathcal{D}}$

Table 1. Workflow of PINNs for data-driven optical soliton solutions of the NLSE in Kerr and non-Kerr law media.

Step 1. Construct the neural network $NN(x, t; \mathbf{W}, \mathbf{b})$, initialize the bias $\mathbf{b}$ to zero and the weights $\mathbf{W}$ using Glorot uniform initialization, and select the nonlinear activation function σ.

Step 2. Generate the datasets $\left\{x_{\mathcal{R}}^j, t_{\mathcal{R}}^j\right\}_{j=1}^{N_{\mathcal{R}}}$, $\left\{x_{\mathcal{I}}^j, t_0; \mathcal{I}_u^j, \mathcal{I}_v^j\right\}_{j=1}^{N_{\mathcal{I}}}$, and $\left\{x_{\mathcal{B}}^j, t_{\mathcal{B}}^j; \mathcal{B}_u^j, \mathcal{B}_v^j\right\}_{j=1}^{N_{\mathcal{B}}}$ used for training the PDE residuals $\mathcal{R}(1)$ and $\mathcal{R}(2)$, initial conditions $\mathcal{I}(u)$ and $\mathcal{I}(v)$, and boundary conditions $\mathcal{B}(u)$ and $\mathcal{B}(v)$, respectively.

Step 3. Design the loss function (2.6) to fit $\mathcal{R}(1)$, $\mathcal{R}(2)$, $\mathcal{I}(u)$, $\mathcal{I}(v)$, $\mathcal{B}(u)$, and $\mathcal{B}(v)$.

Step 4. Update the variables $\boldsymbol{\theta}$ by minimizing the loss (2.6) using the Adam optimizer for m iterations.

Step 5. Further refine $\boldsymbol{\theta}$ using the L-BFGS optimizer. The optimization terminates when $\max\{|\text{proj } g_i| i = 1, ..., n\} \leq 1 \times 10^{-8}$, where $|\text{proj } g_i|$ denotes the i-th component of the projected gradient, or when the maximum iteration M is reached.

Step 6. Save the predicted solution and compute the $\mathbb{L}^2$-norm errors δ between the learned $\{\hat{\mathbf{u}}, \hat{\mathbf{v}}\} = \{\hat{u}_i, \hat{v}_i\}_{i=1}^p$ and exact solutions. Here $\hat{\mathbf{u}}, \hat{\mathbf{v}} \in \mathbb{R}^p$, with p being the number of sampled points in the computational domain.

is defined as follows:

$$\begin{cases} \mathcal{L}_{\mathcal{D}(u)}(\boldsymbol{\theta}; \boldsymbol{\lambda}) = \frac{1}{N_{\mathcal{D}}} \sum_{j=1}^{N_{\mathcal{D}}} [\hat{u}(x_{\mathcal{D}}^j, t') - u(x_{\mathcal{D}}^j, t')]^2, \\ \mathcal{L}_{\mathcal{D}(v)}(\boldsymbol{\theta}; \boldsymbol{\lambda}) = \frac{1}{N_{\mathcal{D}}} \sum_{j=1}^{N_{\mathcal{D}}} [\hat{v}(x_{\mathcal{D}}^j, t') - v(x_{\mathcal{D}}^j, t')]^2, \end{cases} \tag{2.9}$$

where $N_{\mathcal{D}}$ represents the number of observations. Subsequently, we aim to minimize the total loss expressed as $\min_{\theta;\lambda} \mathcal{L}(\boldsymbol{\theta}; \boldsymbol{\lambda})$ with

$$\begin{aligned} \mathcal{L}(\boldsymbol{\theta}; \boldsymbol{\lambda}) = w_{\mathcal{R}} \left[\mathcal{L}_{\mathcal{R}(1)}(\boldsymbol{\theta}; \boldsymbol{\lambda}) + \mathcal{L}_{\mathcal{R}(2)}(\boldsymbol{\theta}; \boldsymbol{\lambda})\right] + w_{\mathcal{I}} \left[\mathcal{L}_{\mathcal{I}(u)}(\boldsymbol{\theta}; \boldsymbol{\lambda}) + \mathcal{L}_{\mathcal{I}(v)}(\boldsymbol{\theta}; \boldsymbol{\lambda})\right] + \\ w_{\mathcal{B}} \left[\mathcal{L}_{\mathcal{B}(u)}(\boldsymbol{\theta}; \boldsymbol{\lambda}) + \mathcal{L}_{\mathcal{B}(v)}(\boldsymbol{\theta}; \boldsymbol{\lambda})\right] + w_{\mathcal{D}} \left[\mathcal{L}_{\mathcal{D}(u)}(\boldsymbol{\theta}; \boldsymbol{\lambda}) + \mathcal{L}_{\mathcal{D}(v)}(\boldsymbol{\theta}; \boldsymbol{\lambda})\right]. \end{aligned} \tag{2.10}$$

The coefficients $\boldsymbol{\lambda} = (\lambda_1, \lambda_2, \mu, \nu)$ do not serve as input or output elements within the network. Consequently, they are exclusively parameters that require optimization, similar to the parameters $\boldsymbol{\theta} = \{\mathbf{W}_j, \mathbf{b}_j\}_{j=1}^N$. As a result, the expressions for $\mathcal{L}_{\mathcal{R}(1)}(\boldsymbol{\theta}; \boldsymbol{\lambda})$, $\mathcal{L}_{\mathcal{R}(2)}(\boldsymbol{\theta}; \boldsymbol{\lambda})$, $\mathcal{L}_{\mathcal{I}(u)}(\boldsymbol{\theta}; \boldsymbol{\lambda})$, $\mathcal{L}_{\mathcal{I}(v)}(\boldsymbol{\theta}; \boldsymbol{\lambda})$, $\mathcal{L}_{\mathcal{B}(u)}(\boldsymbol{\theta}; \boldsymbol{\lambda})$, and $\mathcal{L}_{\mathcal{B}(v)}(\boldsymbol{\theta}; \boldsymbol{\lambda})$ are identical to those defined in (2.6). Furthermore, $w_{\mathcal{R}}$, $w_{\mathcal{I}}$, $w_{\mathcal{B}}$, and $w_{\mathcal{O}}$ represent the weights associated with the respective components.

Subsequently, we can implement the optimization strategies delineated in Section 2.1 to minimize the loss function (2.6), thus determining the unknown coefficients for Eq (2.8). The key steps involved in employing PINNs to ascertain the parameters of Eq (2.8) are summarized in Table 2.

2.2. Exploring data-driven optical soliton solutions through DeepONet

In forward problems of PDEs, the goal is to determine the solution from given inputs, such as initial conditions. This input-to-solution relation defines a solution operator between different function spaces, so the forward problem is naturally an operator-learning problem. Unlike function regression, DeepONet was developed to learn the solution operator that maps input functions to output solution functions. After one offline training stage, it can predict solutions for unseen initial conditions without

Table 2. Workflow of PINNs for coefficient identification in the NLSE under Kerr and non-Kerr nonlinearities.

Step 1. Build the neural network $NN(x, t; \mathbf{W}, \mathbf{b})$, initialize trainable parameters $\boldsymbol{\theta} = \{\mathbf{W}_i, \mathbf{b}_i\}_{i=1}^{N}$, and select the activation function σ. The unknown parameters $\boldsymbol{\lambda} = \{\lambda_1, \lambda_2, \cdots\}$ are treated as additional learnable parameters but are not part of the network's input or output.

Step 2. Apart from the residual, initial, and boundary point sets outlined in Table 1, incorporate an observation dataset $\left\{x_{\mathcal{D}}^j, t'; \mathcal{D}_u^j, \mathcal{D}_v^j\right\}_{j=1}^{N_{\mathcal{D}}}$ at a fixed $t = t'$. This dataset provides supervised data for enhancing the accuracy of the identified parameters $\boldsymbol{\lambda}$.

Step 3. Define the loss function $\mathcal{L}(\boldsymbol{\theta}; \boldsymbol{\lambda})$ as in Eq (2.10), which includes terms for the PDE residuals $\mathcal{R}(1)$ and $\mathcal{R}(2)$, initial conditions $\mathcal{I}(u)$ and $\mathcal{I}(v)$, boundary conditions $\mathcal{B}(u)$ and $\mathcal{B}(v)$, and observation data $\mathcal{D}(u)$ and $\mathcal{D}(v)$.

Step 4. Update the variables $\boldsymbol{\theta}$ and $\boldsymbol{\lambda}$ by minimizing Eq (2.10) using the Adam optimizer.

Step 5. Further refine the variables $\boldsymbol{\theta}$ and $\boldsymbol{\lambda}$ by minimizing Eq (2.10) using the L-BFGS optimizer.

Step 6. Collect the estimated coefficient set $\boldsymbol{\lambda}$ and evaluate the accuracy of the identification using the relative error.

retraining, which is particularly advantageous for large-scale parameter sweeps, repeated forward solves, and real-time control. In this study, we primarily focus on a supervised learning approach, where the training data for the DeepONet are generated from the known analytical solutions of the governing equations. Specifically, the framework of DeepONet that learns the operator that maps initial conditions to the solutions is illustrated in Figure 2, and each step is summarized in Table 3 and detailed as follows.

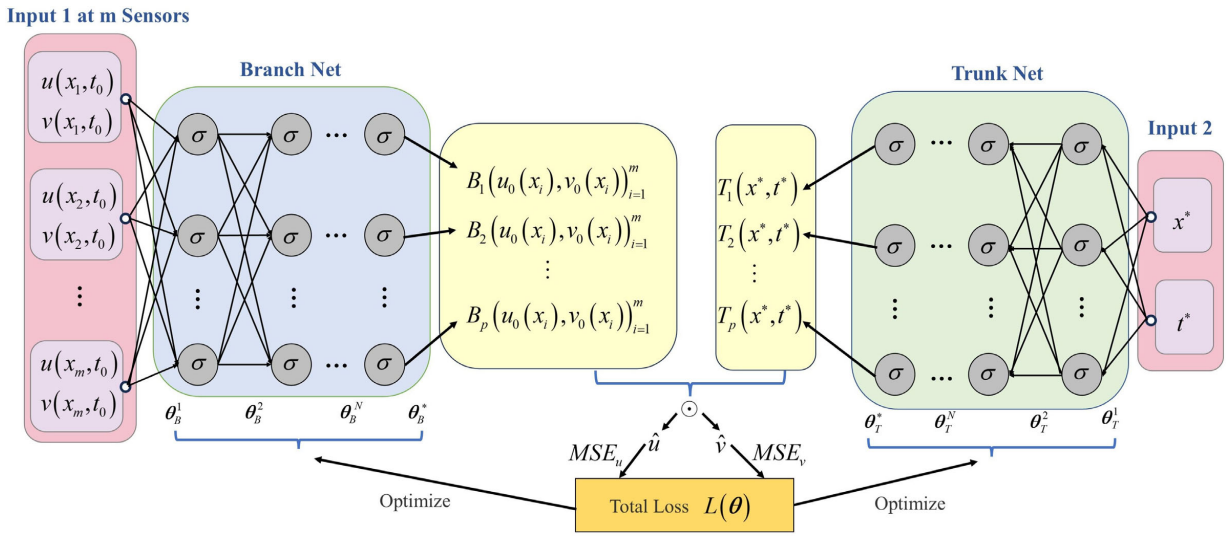


Figure 2. The DeepONet framework for operator learning: approximating the mapping from initial condition samples to the solution space.

Table 3. DeepONet framework for learning the operator that maps initial conditions $u_0(x)$ and $v_0(x)$ to the solutions $\hat{u}(x,t)$ and $\hat{v}(x,t)$.

Step 1. Fix m sensor locations uniformly distributed over $[x_0, x_1]$. For each parameter Λ, evaluate the initial conditions $u_0(x)$, $v_0(x)$ at these locations and at time $t = t_0$; concatenate the real and imaginary parts to form the branch input vector $\mathbf{U}$ as in Eq (2.11).
Step 2. Draw D_Λ distinct values of Λ from $[\Lambda_{\min}, \Lambda_{\max}]$. For each Λ, randomly sample Q spatiotemporal points $\{(x_j^*, t_j^*)\}_{j=1}^Q$, from $[x_0, x_1]\times[t_0, t_1]$, and compute the reference solutions $\left\{u(x_j^*, t_j^*), v(x_j^*, t_j^*)\right\}_{j=1}^Q$. This yields $S = D_\Lambda \times Q$ samples. Split the dataset by Λ: D_Λ^{train} values (giving $M = D_\Lambda^{\text{train}} \times Q$ training samples) and the remaining D_Λ^{test} values for testing. Each sample is the tuple $\left\{\mathbf{U}; (x_j^*, t_j^*); [u(x_j^*, t_j^*), v(x_j^*, t_j^*)]\right\}$. In the network architecture, a generic query point is represented by the vector $\mathbf{Y} = [x^*, t^*]^T$.
Step 3. The predictions $[\hat{u}(\mathbf{U}, \mathbf{Y}), \hat{v}(\mathbf{U}, \mathbf{Y})]$ are obtained via $\mathbf{W}(\mathbf{B}(\mathbf{U}) \odot \mathbf{T}(\mathbf{Y})) + \mathbf{b}$. Here, $\mathbf{B}(\mathbf{U})$ (see Eq (2.12)) and $\mathbf{T}(\mathbf{Y})$ (see Eq (2.14)) are the branch and trunk feature vectors, $\odot$ denotes the Hadamard product, $\mathbf{W} \in \mathbb{R}^{2\times p}$ is a weight matrix, and $\mathbf{b} = [b_u, b_v]^T \in \mathbb{R}^2$ is a bias vector.
Step 4. Optimize all parameters $\boldsymbol{\theta} = \{\boldsymbol{\theta}_B, \boldsymbol{\theta}_T, \boldsymbol{\theta}_O\}$ by minimizing the loss function (2.18) over the M training samples. Use the Adam optimizer with a fixed learning rate lr and mini-batch size mb, and train for a prescribed number of epochs.

2.2.1. The generation of training and testing data

We begin by detailing the procedure for constructing the dataset to train and test the DeepONet model. First, D_Λ distinct values of the parameter Λ are randomly drawn from the prescribed interval $[\Lambda_{\min}, \Lambda_{\max}]$. For each such Λ, the initial condition is evaluated at m uniformly distributed sensor locations over the spatial domain $[x_0, x_1]$ at time $t = t_0$. The resulting values of real and imaginary parts $\{u_0(x_i), v_0(x_i)\}_{i=1}^m$ are concatenated to form the input of the branch network. Next, for this fixed initial condition, we randomly sample Q spatiotemporal points $\left\{(x_j^*, t_j^*)\right\}_{j=1}^Q$ from the domain $[x_0, x_1] \times [t_0, t_1]$; the corresponding reference solutions $\left\{u(x_j^*, t_j^*), v(x_j^*, t_j^*)\right\}_{j=1}^Q$ are computed using the exact solution formula. Each pair (x_j^*, t_j^*) together with the shared branch input constitutes one complete training sample. Repeating this procedure for all D_Λ values of Λ yields a total of $S = D_\Lambda \times Q$ samples. In contrast to using a single query point per initial condition, the introduction of Q enables the optimization of training performance with a finite number of initial conditions by tuning the per-condition spatiotemporal sampling density. Since Λ is drawn from a compact interval and the mapping $\Lambda \mapsto (u_0, v_0)$ is continuous, the resulting set of initial-condition functions forms a compact subset of $L^2([x_0, x_1]) \times L^2([x_0, x_1])$, satisfying the compactness requirement of the universal approximation theorem for operators. To assess the model's ability to generalize to unseen initial conditions, the dataset is split by Λ into a training set and a test set. Specifically, D_Λ^{train} distinct values of Λ (together with all their associated Q samples) are randomly selected for training, while the remaining $D_\Lambda^{\text{test}} = D_\Lambda - D_\Lambda^{\text{train}}$ values are reserved for testing. This ensures that no initial condition appearing in the training set is ever encountered during testing, thereby providing an unbiased evaluation of the operator's generalization capability.

2.2.2. The architecture of neural networks

The DeepONet architecture comprises two parallel subnetworks: a branch network and a trunk network. In this work, both are implemented as an FNN with N hidden layers, each containing p neurons and a specific activation function σ. Specifically, the branch network encodes the initial condition discretized at m fixed sensor locations over $[x_0, x_1]$ at time $t = t_0$. Its input is a $2m$-dimensional vector formed by concatenating the real and imaginary parts of the initial condition at those locations:

$$\mathbf{U} = [u_0(x_1), v_0(x_1), u_0(x_2), v_0(x_2), \ldots, u_0(x_m), v_0(x_m)]^T \in \mathbb{R}^{2m}. \tag{2.11}$$

After passing through N hidden layers, the branch network outputs a p-dimensional feature vector

$$\mathbf{B}(\mathbf{U}) = [B_1(\mathbf{U}), B_2(\mathbf{U}), \ldots, B_p(\mathbf{U})]^T \in \mathbb{R}^p, \tag{2.12}$$

which captures essential characteristics from the input initial condition. Additionally, the trunk network takes a spatiotemporal query point (x^*, t^*) as input, which is represented by a two-dimensional coordinate vector

$$\mathbf{Y} = [x^*, t^*]^T \in \mathbb{R}^2. \tag{2.13}$$

After processing through N hidden layers, the network outputs a p-dimensional feature vector

$$\mathbf{T}(\mathbf{Y}) = [T_1(\mathbf{Y}), T_2(\mathbf{Y}), \ldots, T_p(\mathbf{Y})]^T \in \mathbb{R}^p. \tag{2.14}$$

The final output of DeepONet is obtained by first computing the element-wise (Hadamard) product of the two feature vectors (2.12) and (2.14), followed by a linear transformation. For the complex-valued Eq (1.1), the model predicts both the real and imaginary parts as

$$\begin{bmatrix} \hat{u}(\mathbf{U}, \mathbf{Y}) \\ \hat{v}(\mathbf{U}, \mathbf{Y}) \end{bmatrix} = \mathbf{W}(\mathbf{B}(\mathbf{U}) \odot \mathbf{T}(\mathbf{Y})) + \mathbf{b}, \tag{2.15}$$

where $\odot$ denotes the Hadamard product, $\mathbf{W} \in \mathbb{R}^{2\times p}$ is a weight matrix, and $\mathbf{b} = [b_u, b_v]^T \in \mathbb{R}^2$ is a bias vector. Equivalently, the outputs can be rewritten as

$$\hat{u}(\mathbf{U}, \mathbf{Y}) = \sum_{i=1}^{p} w_{u,i} \cdot B_i(\mathbf{U}) T_i(\mathbf{Y}) + b_u, \tag{2.16}$$

$$\hat{v}(\mathbf{U}, \mathbf{Y}) = \sum_{i=1}^{p} w_{v,i} \cdot B_i(\mathbf{U}) T_i(\mathbf{Y}) + b_v, \tag{2.17}$$

where $w_{u,i}$ and $w_{v,i}$ are the entries of $\mathbf{W}$ corresponding to the real and imaginary outputs, respectively.

2.2.3. The composition of the loss function

During training, all network parameters $\boldsymbol{\theta} = \{\boldsymbol{\theta}_B, \boldsymbol{\theta}_T, \boldsymbol{\theta}_O\}$ are optimized to minimize the discrepancy between the model predictions $\{\hat{u}, \hat{v}\}$ and the reference solutions $\{u, v\}$. Here, $\boldsymbol{\theta}_B = \{\boldsymbol{\theta}_B^1, ..., \boldsymbol{\theta}_B^N\}$ and $\boldsymbol{\theta}_T = \{\boldsymbol{\theta}_T^1, ..., \boldsymbol{\theta}_T^N\}$ denote the weights and biases of the branch and trunk networks, respectively, while $\boldsymbol{\theta}_O = \{\mathbf{W}, \mathbf{b}\}$ represents the parameters of the output layer. As described in Section 2.2.1, the training set consists of $M = D_\Lambda^{\text{train}} \times Q$ samples, where each sample is associated with one of the D_Λ^{train} parameter

values and one of the Q spatiotemporal points drawn for that parameter. The loss function is defined as the mean squared error (MSE) averaged over the M training samples:

$$\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{M}\sum_{i=1}^{M}\left[\left(\hat{u}^{(i)} - u^{(i)}\right)^2 + \left(\hat{v}^{(i)} - v^{(i)}\right)^2\right], \tag{2.18}$$

where $\left\{\hat{u}^{(i)}, \hat{v}^{(i)}\right\}$ and $\left\{u^{(i)}, v^{(i)}\right\}$ correspond to the predicted and reference solutions for the i-th sample. Next, we employ the Adam optimizer with a fixed learning rate lr to minimize the loss (2.18). The model is trained for a prescribed number of epochs, and gradient updates are computed using mini-batches of size mb.

In the following sections, we conduct numerical experiments to investigate the forward and inverse problems of Eq (1.1) via the proposed deep learning frameworks. All numerical implementations were developed in Python 3.8 with TensorFlow 2.9.0, and computations were accelerated using an NVIDIA GeForce RTX 3090 GPU.

3. Forward problems

In this section, we employ PINNs and DeepONet to explore data-driven solutions of the NLSE with various nonlinearities, including the Kerr, power, parabolic, and logarithmic laws. Several factors that may affect model performance are also discussed (see Table A1 in Appendix A for a quick reference). Except for the factor under investigation, all other hyperparameters are kept fixed across all experimental configurations.

3.1. Data-driven solution of the NLSE with Kerr law nonlinearity

For the case of Kerr law nonlinearity, we have $\mathcal{F}(|\psi|^2) = |\psi|^2$. Thus, Eq (1.1) reduces to the standard NLSE

$$i\psi_t + \frac{1}{2}\psi_{xx} + |\psi|^2\psi = 0, \tag{3.1}$$

which is integrable, and its one-soliton solution can be written as follows [5]:

$$\psi(x,t) = A\,\mathrm{sech}[B(x-kt)]e^{i(kx+wt)}, \tag{3.2}$$

where A, B, k, w are real parameters, with the conditions $A = B$ and $w = \frac{A^2-k^2}{2}$ being satisfied.

Remark 1. *It is crucial to emphasize that Eq (3.1) also admits other exact solutions, such as multi-soliton, breather, rogue-wave, periodic, and rational solutions (see Refs. [8, 9, 46] and the references therein). Nevertheless, this study focuses on the fundamental one-soliton solution because (i) it is the basic nonlinear mode in optical fibers and communications, and captures the essential dispersion-nonlinearity balance; (ii) closed-form one-soliton solutions are available for all four considered nonlinearities, enabling a fair comparison without additional effects such as soliton collisions or phase shifts; and (iii) it decays rapidly at the boundaries, yielding simple and well-posed initial-boundary data, so that the observed errors mainly reflect sensitivity to hyperparameters, the optimization scheme, and noise robustness.*

3.1.1. PINNs method

First, we apply the PINNs framework to Eq (3.1). The computational domain $[x_0, x_1]$ and $[t_0, t_1]$ of Eq (3.1) are selected as $[-10, 10]$ and $[-5, 5]$, respectively. Choosing the parameters $A = 1$, $B = 1$, and $k = 0.5$ in solution (3.2) yields the corresponding initial and boundary conditions. Specifically, the initial conditions at $t_0 = -5$ are given by

$$\begin{cases} \mathcal{I}_u(x, -5) = \operatorname{sech}(x + 2.5)\cos(0.5x - 1.875), \\ \mathcal{I}_v(x, -5) = \operatorname{sech}(x + 2.5)\sin(0.5x - 1.875). \end{cases} \tag{3.3}$$

The boundary conditions are obtained by evaluating Eq (3.2) at $x_0 = -10$ and $x_1 = 10$:

$$\begin{cases} \mathcal{B}_u(-10, t) = \operatorname{sech}(-10 - 0.5t)\cos(-5 + 0.375t), \\ \mathcal{B}_u(10, t) = \operatorname{sech}(10 - 0.5t)\cos(5 + 0.375t), \\ \mathcal{B}_v(-10, t) = \operatorname{sech}(-10 - 0.5t)\sin(-5 + 0.375t), \\ \mathcal{B}_v(10, t) = \operatorname{sech}(10 - 0.5t))\sin(5 + 0.375t). \end{cases} \tag{3.4}$$

In this work, we employ the hyperbolic tangent (tanh) as the activation function. For training, we randomly sampled $N_{\mathcal{R}} = 10,000$ collocation points for the residual, $N_{\mathcal{I}} = 200$ points for the initial conditions, and $N_{\mathcal{B}} = 400$ points for the boundary conditions. The optimization process consists of two stages. At first, the model parameters are optimized for 6000 iterations using the Adam optimizer. Subsequently, the L-BFGS optimizer is applied to refine the solution until convergence is achieved. Table 4 illustrates the impact of different network architectures on the accuracy of the learned solution $\left|\hat{\psi}(x, t)\right|$ for Eq (3.1). The results indicate that the prediction error generally decreases as the network depth and width increase. For a fixed width of 45 neurons, a depth of 4 yields slightly better accuracy than a depth of 3. However, the training time of the former is over ten seconds longer. Consequently, we adopt a network with three hidden layers of 45 neurons each for all subsequent numerical experiments.

Table 4. $\mathbb{L}^2$-norm error of $\left|\hat{\psi}(x, t)\right|$ for Eq (3.1) solved by PINNs with varying network depths and neuron counts per layer.

Layers	Number of neurons per hidden layer				
	20	30	40	45	50
2	1.488×10^{-2}	6.475×10^{-3}	4.245×10^{-3}	1.701×10^{-3}	1.425×10^{-3}
3	3.240×10^{-3}	3.711×10^{-3}	2.315×10^{-3}	1.123×10^{-3}	1.984×10^{-3}
4	8.864×10^{-3}	2.544×10^{-3}	1.416×10^{-3}	1.071×10^{-3}	2.420×10^{-3}
5	7.331×10^{-3}	4.405×10^{-3}	2.181×10^{-3}	1.766×10^{-3}	7.485×10^{-4}

Figure 3 shows the real and imaginary parts of the data-driven solution, $\hat{u}(x, t)$ and $\hat{v}(x, t)$, for Eq (3.1). The network training required approximately 96.03 seconds, with the optimal model concluding at the 14,846th iteration. The $\mathbb{L}^2$-norm error for $\hat{u}(x, t)$ is 2.653×10^{-3} and for $\hat{v}(x, t)$ is 2.760×10^{-3}. The 3D profiles illustrating $\hat{u}(x, t)$ and $\hat{v}(x, t)$ are displayed in Figure 3(a1),(a2), while the corresponding density diagrams are shown in Figure 3(b1),(b2). Additionally, the absolute errors between the predictions and exact values are exhibited in Figure 3(c1),(c2). To visually demonstrate the efficacy of PINNs, we present a comparative analysis of the target and predicted values of $\hat{u}(x, t)$ and $\hat{v}(x, t)$ at $t = -3$ in Figure 3(d1),(d2). Similarly, the comparisons at $t = 3$ are shown in Figure 3(e1),(e2).

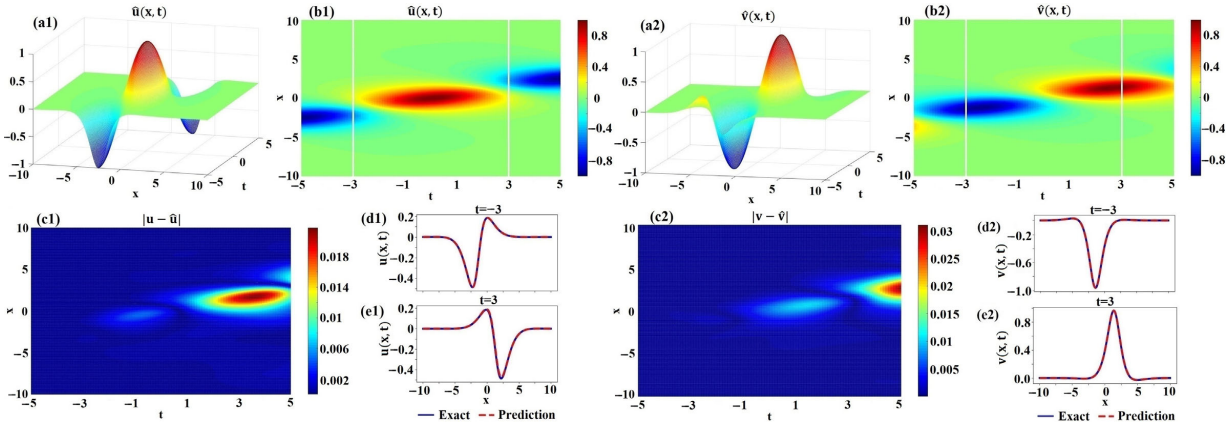


Figure 3. Data-driven real part $\hat{u}(x,t)$ and imaginary part $\hat{v}(x,t)$ of Eq (3.1) with Kerr law nonlinearity. (a1),(a2) 3D visualizations of $\hat{u}(x,t)$ and $\hat{v}(x,t)$. (b1),(b2) Density diagrams of $\hat{u}(x,t)$ and $\hat{v}(x,t)$. (c1),(c2) Error dynamics of $|u-\hat{u}|$ and $|v-\hat{v}|$. (d1),(d2) Comparisons between the exact solution and prediction at $t=-3$. (e1),(e2) Comparisons between the exact solution and prediction at $t=3$.

Regarding the modulus $\left|\hat{\psi}(x,t)\right|$ of the learned solution to Eq (3.1), the 3D visualization of $\left|\hat{\psi}(x,t)\right|$ is presented in Figure 4(a1) with an $\mathbb{L}^2$-norm error of 1.123×10^{-3}. The predicted one-soliton solution exhibits a bell-shaped profile, and its propagation speed is directly proportional to its amplitude. The dynamic error related to $\left|\hat{\psi}(x,t)\right|$ is shown in Figure 4(b1). Comparisons between the exact and numerical solution at times $t=-3$ and $t=3$ are displayed in Figure 4(c1), from which we can observe that the soliton propagates in the positive direction along the x-axis as the temporal variable t evolves. Specifically, the approximation represented by the full line and the dotted line in Figure 4(c1) indicates the efficacy of PINNs in exploring Eq (3.1).

3.1.2. DeepONet method

We begin by constructing the dataset for the DeepONet method. For a fixed value of $k=0.5$ in Eq (3.2), we randomly sample 200 distinct amplitudes A from the range $0.1<A<2$. For each such A, we then sample $Q=50$ random spatiotemporal points to help the network better learn the solution associated with that initial condition. This yields a total of 10,000 samples originating from 200 different initial conditions. We randomly select $D_\Lambda^{\text{train}}=160$ of these A values for training and retain the remaining $D_\Lambda^{\text{test}}=40$ for testing, resulting in 8000 training samples and 2000 test samples. The two sets are strictly separated by A, thus preventing any data leakage. Unless otherwise specified, the data sample size and split ratio in the subsequent experiments remain the same as above. To encode the initial conditions $u(x,-5)$ and $v(x,-5)$ from Eq (3.3) into the branch network, we distribute 50 sensor locations uniformly over the spatial domain $[-10,10]$. The branch network takes an input layer of 100 neurons, while the trunk network input layer has 2 neurons. In this work, we examine the impact of different architectures of the branch and trunk networks on the model's prediction accuracy. After training for 100,000 iterations using the Adam optimizer with a learning rate of 0.0005, the predictions $\hat{u}(x,t)$ and $\hat{v}(x,t)$ are obtained by computing the Hadamard product of the output feature vectors from the branch and trunk networks.

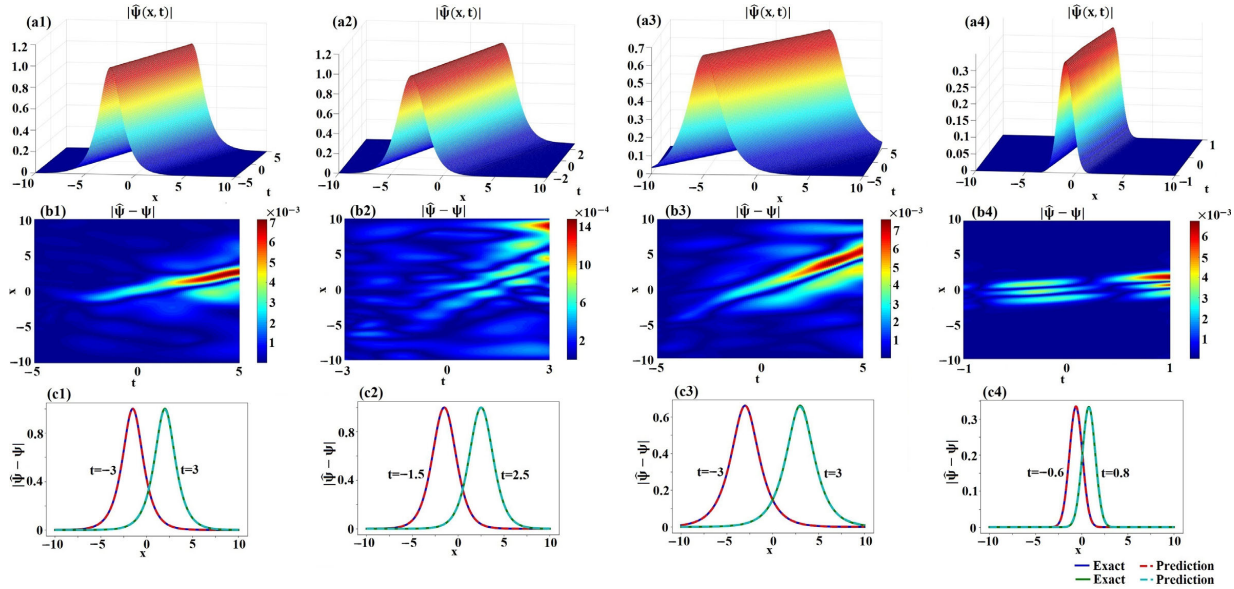


Figure 4. The modulus $\left|\hat{\psi}(x,t)\right|$ of the data-driven solution for NLSEs in both Kerr and non-Kerr law media. The first row: 3D visualizations of $\left|\hat{\psi}(x,t)\right|$. The second row: Absolute error between $|\psi(x,t)|$ and $\left|\hat{\psi}(x,t)\right|$. The third row: Comparisons between $|\psi(x,t)|$ and $\left|\hat{\psi}(x,t)\right|$ at distinct times. Columns one to four correspond to the NLSE with Kerr, power, parabolic, and logarithmic nonlinearities.

First of all, we adopted identical architectures for both the branch and trunk networks. The overall $\mathbb{L}^2$-norm error for $\left|\hat{\psi}(x,t)\right|$, averaged over 2000 test samples, is summarized in Table 5. Our experiments indicate that a three-layer FNN with 128 neurons per layer represents an optimal choice. In fact, the final feature layers of both the branch and trunk networks have dimensions 128×2, since DeepONet has two output components. This architecture completed training in 475.70 seconds, achieving $\mathbb{L}^2$-norm errors of 3.624×10^{-2} for $\hat{u}(x,t)$ and 3.038×10^{-2} for $\hat{v}(x,t)$ on the test dataset. We subsequently fixed the branch network architecture and investigated the impact of reducing the trunk network's complexity. As demonstrated in [31], employing complex architectures for the branch network leads to significant performance gains, while keeping the trunk network simpler remains most effective. However, this conclusion primarily applies to physics-informed DeepONet. The results in Table 6 reveal that for a purely data-driven DeepONet, simplifying the trunk network does not yield obvious benefits. Therefore, in accordance with the common practice in most DeepONet literature, this study maintains identical architectures for both sub-networks (with the exception of their input layers).

As previously mentioned, once trained, DeepONet can provide real-time predictions for scenarios with unseen parameters. To demonstrate its performance, we select three representative cases from the test dataset. Figure 5 illustrates DeepONet's predicted solutions to Eq (3.1) with Kerr-law nonlinearity. Specifically, in Figure 5(a1)–(a3), the exact and learned real parts are compared at $t = 0$ for $A = 0.3$, $A = 1$, and $A = 1.7$, respectively. The corresponding imaginary parts are shown in Figure 5(b1)–(b3). Additionally, Figure 5(c) displays the training and testing loss curves throughout the optimization process. For a more detailed analysis, Figure 6 focuses on the case where $A = 1$, exhibiting both the data-driven solutions and the associated error dynamics. The exact and predicted real parts are depicted

Table 5. $\mathbb{L}^2$-norm error of $\left|\hat{\psi}(x,t)\right|$ (evaluated on 2000 test samples) for Eq (3.1) solved by DeepONet with varying network depths and neuron counts per layer.

Layers	Number of neurons per layer				
	16	32	64	128	256
2	1.302×10^{-1}	6.452×10^{-2}	6.143×10^{-2}	5.108×10^{-2}	5.522×10^{-2}
3	5.971×10^{-2}	4.117×10^{-2}	3.009×10^{-2}	2.443×10^{-2}	5.487×10^{-2}
4	4.294×10^{-2}	3.691×10^{-2}	2.585×10^{-2}	3.610×10^{-2}	6.659×10^{-2}
5	4.046×10^{-2}	2.730×10^{-2}	3.604×10^{-2}	4.109×10^{-2}	1.033×10^{-1}

Table 6. $\mathbb{L}^2$-norm error of $\left|\hat{\psi}(x,t)\right|$ (evaluated on 2000 test samples) for Eq (3.1) solved by DeepONet with different branch networks and trunk networks.

Branch net	Trunk net					
	[32] ∗ 2 + [64]	[32] ∗ 2 + [128]	[64] ∗ 2 + [64]	[64] ∗ 2 + [128]	[128] ∗ 2 + [32]	[128] ∗ 2 + [128]
[64] ∗ 3	6.834×10^{-2}	–	3.006×10^{-2}	–	3.094×10^{-2}	–
[128] ∗ 3	–	9.385×10^{-2}	–	5.012×10^{-2}	–	2.440×10^{-2}

**Note*: The symbol "–" indicates untested configurations due to mismatched branch and trunk output dimensions.

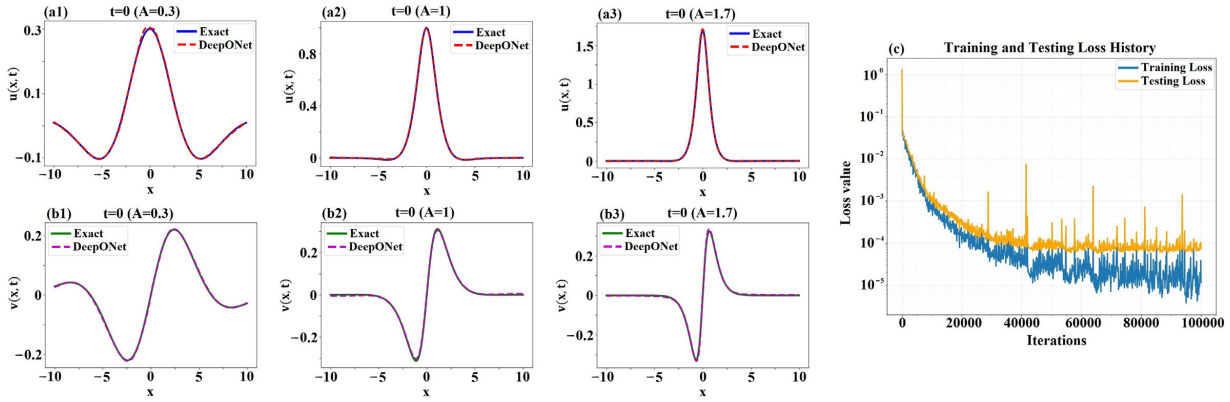


Figure 5. The results of DeepONet in solving Eq (3.1) with Kerr law nonlinearity. (a1),(b1) $\hat{u}(x,t)$ and $\hat{v}(x,t)$ for $A = 0.3$ at $t = 0$. (a2),(b2) $\hat{u}(x,t)$ and $\hat{v}(x,t)$ for $A = 1$ at $t = 0$. (a3),(b3) $\hat{u}(x,t)$ and $\hat{v}(x,t)$ for $A = 1.7$ at $t = 0$. (c) The curve of training and testing loss.

as density plots in Figure 6(a1),(a2), with the absolute error $|u - \hat{u}|$ shown in Figure 6(a3). Similarly, in Figure 6(b1),(b2), the exact and learned imaginary part are compared, while Figure 6(b3) visualizes the error $|v - \hat{v}|$. Furthermore, the exact and predicted moduli are displayed in Figure 6(c1),(c2), with the corresponding error $\left|\psi(x,t) - \hat{\psi}(x,t)\right|$ shown in Figure 6(c3). In particular, the real-time inference efficiency of DeepONet and its network parameter count are further provided in Appendix B (see Tables B1 and B4).

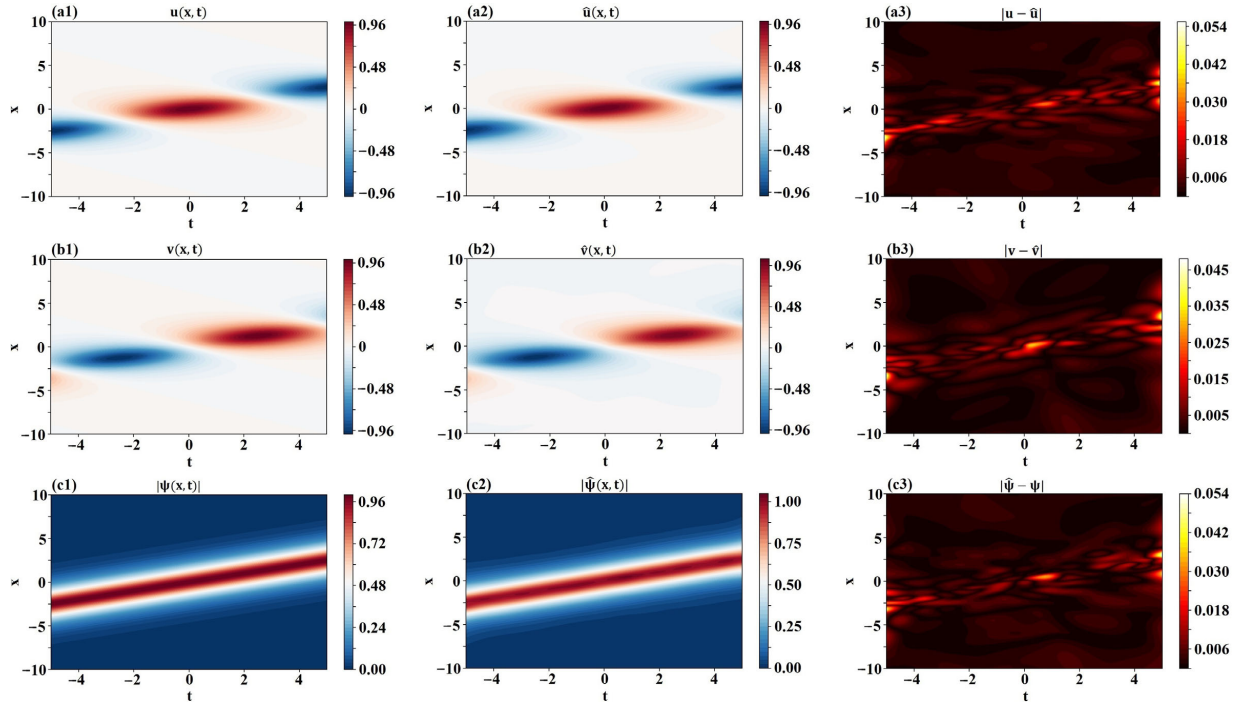


Figure 6. Data-driven solutions and error dynamics of Eq (3.1) with $A = 1$ via DeepONet. (a1)–(c1) Density diagrams of exact $u(x,t)$, $v(x,t)$, and $|\psi(x,t)|$. (a2)–(c2) Density diagrams of predicted $\hat{u}(x,t)$, $\hat{v}(x,t)$, and $\left|\hat{\psi}(x,t)\right|$. (a3)–(c3) Error dynamics of $|u-\hat{u}|$, $|v-\hat{v}|$, and $\left|\psi(x,t)-\hat{\psi}(x,t)\right|$.

3.2. Data-driven solution of the NLSE with power law nonlinearity

Given that the power law nonlinearity $\mathcal{F}(|\psi|^2) = (|\psi|^2)^\mu$, Eq (1.1) takes the form

$$i\psi_t + \frac{1}{2}\psi_{xx} + |\psi|^{2\mu}\psi = 0. \tag{3.5}$$

The solution of Eq (3.5) is given by [5]

$$\psi(x,t) = \frac{A}{\cosh^{1/\mu}(B(x-kt))} e^{i(kx+wt)}, \tag{3.6}$$

where $B = A^\mu \sqrt{\frac{2\mu^2}{1+\mu}}$ and $w = \frac{B^2}{2\mu^2} - \frac{k^2}{2}$. To avoid wave collapse and self-focusing singularities, the condition $0 < \mu < 2$ must be satisfied [6, 7].

3.2.1. PINNs method

We set $[x_0, x_1]$ and $[t_0, t_1]$ in Eq (3.5) as $[-10, 10]$ and $[-3, 3]$. The initial conditions are established by selecting $A = 1$, $\mu = 0.5$, and $k = 1$; subsequently, $B = 0.57735$ and $w = 0.16667$ are calculated and substituted into Eq (3.6). Similarly, the boundary conditions are derived using the same procedure.

In Section 2.1, we demonstrated that the loss function consists of a synthesis of three components, each assigned a specific weight. The simplest scenario is to set $w_{\mathcal{R}} = w_{\mathcal{I}} = w_{\mathcal{B}} = 1$. In exploring the

impact of weights on loss, we employ different values for $w_{\mathcal{R}}$, $w_{\mathcal{I}}$, and $w_{\mathcal{B}}$. As demonstrated in Table 7, the precision of the learned solution for Eq (3.5) can reach $O(10^{-4})$ when the weights $\{w_{\mathcal{R}}, w_{\mathcal{I}}, w_{\mathcal{B}}\}$ are set to $\{1, 0.5, 0.5\}$, $\{1, 0.5, 0.1\}$, or $\{1, 0.1, 0.01\}$. Based on experience, a term whose loss decreases faster means that it is easier to train, so the weight of this item can be appropriately set smaller. In contrast, for items with high loss values, their weights can be appropriately set larger.

Table 7. Effect of weight distributions $\{w_{\mathcal{R}}, w_{\mathcal{I}}, w_{\mathcal{B}}\}$ of the loss function for solving the NLSE (3.5) with power law nonlinearity.

Weights	$\mathbb{L}^2$-norm errors			Total iterations	Running time (s)
	$u(x,t)$	$v(x,t)$	$\lvert\psi(x,t)\rvert$		
$w_{\mathcal{R}}=1, w_{\mathcal{I}}=1, w_{\mathcal{B}}=1$	1.708×10^{-3}	1.728×10^{-3}	1.226×10^{-3}	13,982	81.56
$w_{\mathcal{R}}=1, w_{\mathcal{I}}=0.5, w_{\mathcal{B}}=0.5$	1.248×10^{-3}	1.192×10^{-3}	8.891×10^{-4}	14,745	88.92
$w_{\mathcal{R}}=1, w_{\mathcal{I}}=0.5, w_{\mathcal{B}}=0.1$	1.047×10^{-3}	1.212×10^{-3}	8.976×10^{-4}	13,931	90.62
$w_{\mathcal{R}}=1, w_{\mathcal{I}}=0.1, w_{\mathcal{B}}=0.01$	9.525×10^{-4}	1.054×10^{-3}	7.056×10^{-4}	13,965	83.32
$w_{\mathcal{R}}=1, w_{\mathcal{I}}=0.1, w_{\mathcal{B}}=0.001$	2.201×10^{-3}	2.602×10^{-3}	2.032×10^{-3}	12,824	75.81
$w_{\mathcal{R}}=1, w_{\mathcal{I}}=0.01, w_{\mathcal{B}}=0.001$	6.136×10^{-3}	6.125×10^{-3}	4.029×10^{-3}	13,792	79.09

Figure 7 displays the real and imaginary components of the data-driven solution to Eq (3.5). When the weights for each component of the loss function are set to $w_{\mathcal{R}}=1$, $w_{\mathcal{I}}=0.1$, and $w_{\mathcal{B}}=0.01$, the training of the network required approximately 83.32 seconds, with the optimal model concluding at the 13,965th iteration. The $\mathbb{L}^2$-norm error for $\hat{u}(x,t)$ is 9.525×10^{-4}, while that for the learned $\hat{v}(x,t)$ is 1.054×10^{-3}. The 3D visualizations of $\hat{u}(x,t)$ and $\hat{v}(x,t)$ are exhibited in Figure 7(a1),(a2), while the corresponding density diagrams are illustrated in Figure 7(b1),(b2). Moreover, the absolute errors between the exact solution and prediction are shown in Figure 7(c1),(c2). Comparisons of prediction accuracy at $t=-1.5$ and $t=1.5$ are presented in Figure 7(d1),(d2) and 7(e1),(e2), respectively.

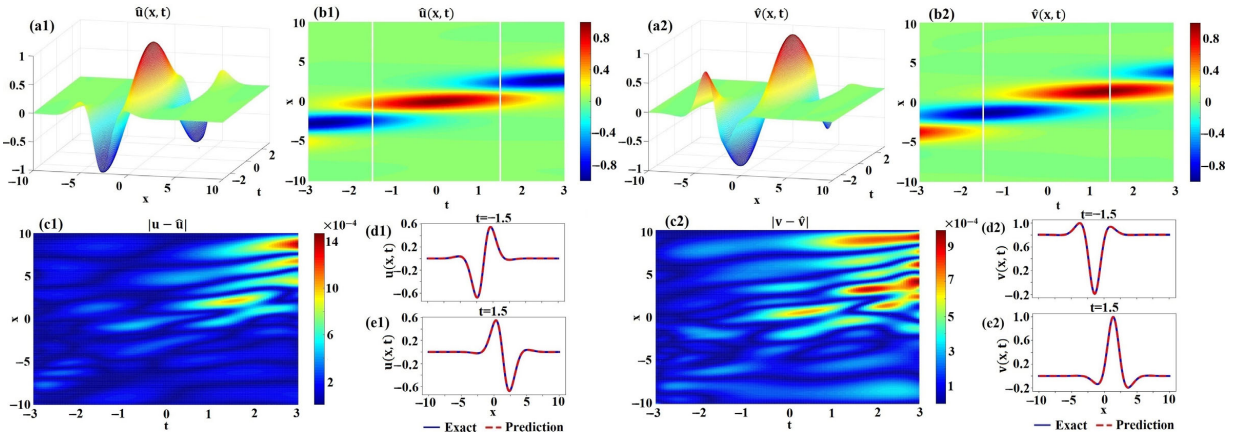


Figure 7. Data-driven real part $\hat{u}(x,t)$ and imaginary part $\hat{v}(x,t)$ of Eq (3.5) with power law nonlinearity. (a1),(a2) 3D visualizations of $\hat{u}(x,t)$ and $\hat{v}(x,t)$. (b1),(b2) Density diagrams of $\hat{u}(x,t)$ and $\hat{v}(x,t)$. (c1),(c2) Error dynamics of $|u-\hat{u}|$ and $|v-\hat{v}|$. (d1),(d2) Comparisons between the exact solution and prediction at $t=-1.5$. (e1),(e2) Comparisons between the exact solution and prediction at $t=1.5$.

Meanwhile, the 3D profile of the modulus $\left|\hat{\psi}(x,t)\right|$ of the data-driven solution to Eq (3.5) is displayed

in Figure 4(a2) with an $\mathbb{L}^2$-norm error of 7.056×10^{-4}. The dynamic error of $\left|\hat{\psi}(x,t)\right|$ is given in Figure 4(b2). Comparisons of exact solution and prediction at $t = -1.5$ and $t = 2.5$ are illustrated in Figure 4(c2), from which we can observe that the soliton propagates in the positive direction along the x-axis as the temporal variable t advances.

3.2.2. DeepONet method

In this part, we extend DeepONet with a three-layer architecture, comprising 128 neurons per layer, to solve Eq (3.5). Sensor locations are uniformly distributed across the spatial domain $[-10, 10]$ at $t = -3$, while the parameter μ in Eq (3.5) is sampled from the interval $[0.1, 1.9]$. We then investigate the influence of training data size S and the number of sensor configurations m on prediction accuracy. Results in Table 8 demonstrate that the prediction accuracy generally improves with larger training datasets. However, increasing the number of sensor locations does not consistently enhance performance. The solution $\left|\hat{\psi}(x,t)\right|$ of Eq (3.5) already achieves satisfactory accuracy with $m = 10$. Furthermore, the sensitivity of DeepONet to the distribution of initial-condition sensor locations is examined by comparing three sampling strategies: equidistant sampling, independent uniform random sampling, and Latin hypercube sampling. As shown in Table B3 in Appendix B, the prediction results are insensitive to the initial-condition sensor locations. For the subsequent study, we therefore employ a dataset of 10,000 samples and set $m = 50$ by equidistant sampling. Under this configuration, training for 100,000 iterations using the Adam optimizer required 335.72 seconds. The resulting $\mathbb{L}^2$-norm errors of $\hat{u}(x,t)$ and $\hat{v}(x,t)$ on the test dataset are 1.053×10^{-2} and 7.425×10^{-3}, respectively.

Table 8. $\mathbb{L}^2$-norm error of $\left|\hat{\psi}(x,t)\right|$ for DeepONet solutions to Eq (3.5) under varying training data sizes and sensor configurations.

Dataset size	Number of sensor locations					
	10	20	50	100	200	300
3000	1.338×10^{-2}	1.035×10^{-2}	1.650×10^{-2}	2.318×10^{-2}	2.123×10^{-2}	2.181×10^{-2}
5000	7.791×10^{-3}	1.017×10^{-2}	8.164×10^{-3}	2.120×10^{-2}	1.501×10^{-2}	4.045×10^{-2}
10,000	7.665×10^{-3}	9.742×10^{-3}	6.156×10^{-3}	1.306×10^{-2}	1.467×10^{-2}	2.369×10^{-2}
15,000	7.236×10^{-2}	8.608×10^{-3}	1.112×10^{-2}	1.137×10^{-2}	2.162×10^{-2}	1.692×10^{-2}

Figure 8 presents the data-driven solutions to Eq (3.5) via DeepONet. Specifically, Figure 8(a1)–(a3) compare the exact and predicted real parts of the solution at $t = 0$ for $\mu = 0.5$, $\mu = 1$, and $\mu = 1.7$, respectively. The corresponding imaginary parts are displayed in Figure 8(b1)–(b3). Furthermore, the training and testing loss curves are shown in Figure 8(c). A detailed analysis for the case where $\mu = 1$ is provided in Figure 9. The density plots of the exact and predicted real parts are exhibited in Figure 9(a1),(a2), with the corresponding error distribution $|u - \hat{u}|$ visualized in Figure 9(a3). Similarly, Figure 9(b1),(b2) compare the exact and learned imaginary parts, while Figure 9(b3) illustrates the error $|v - \hat{v}|$. Additionally, the exact and learned moduli are depicted in Figure 9(c1),(c2), with the error $\left|\psi(x,t) - \hat{\psi}(x,t)\right|$ presented in Figure 9(c3).

3.3. Data-driven solution of the NLSE with parabolic law nonlinearity

For the nonlinearity characterized by the parabolic law, $\mathcal{F}(|\psi|^2) = |\psi|^2 + \nu(|\psi|^2)^2$, Eq (1.1) becomes

$$i\psi_t + \frac{1}{2}\psi_{xx} + \left(|\psi|^2 + \nu\,|\psi|^4\right)\psi = 0. \tag{3.7}$$

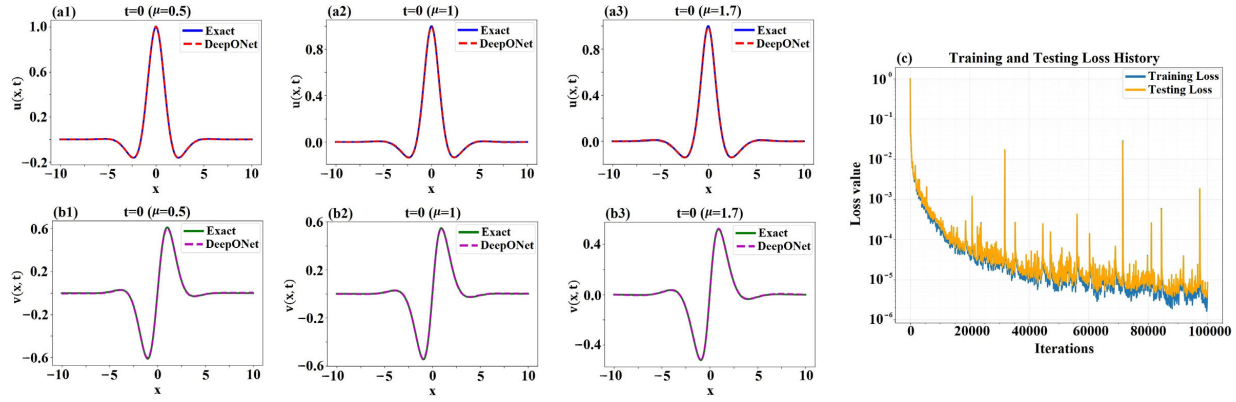


Figure 8. The results of DeepONet in solving Eq (3.5) with power law nonlinearity. (a1),(b1) $\hat{u}(x,t)$ and $\hat{v}(x,t)$ for $\mu = 0.5$ at $t = 0$. (a2),(b2) $\hat{u}(x,t)$ and $\hat{v}(x,t)$ for $\mu = 1$ at $t = 0$. (a3),(b3) $\hat{u}(x,t)$ and $\hat{v}(x,t)$ for $\mu = 1.7$ at $t = 0$. (c) The curve of training and testing loss.

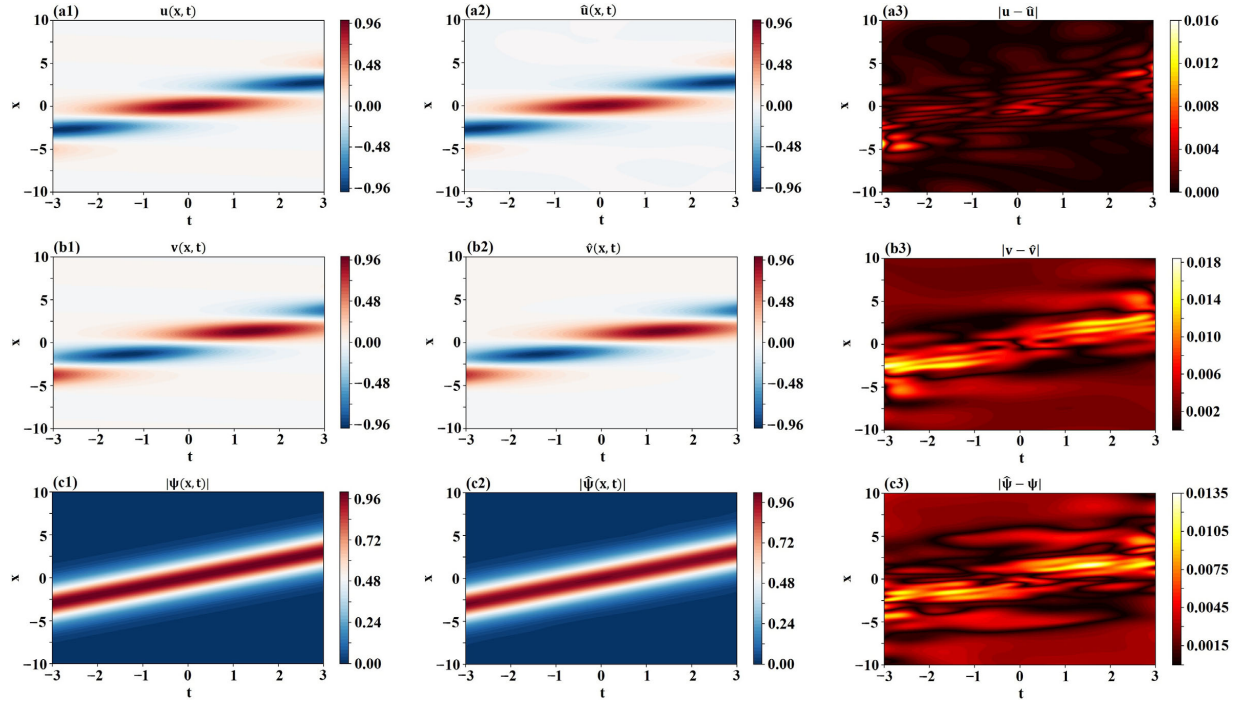


Figure 9. Data-driven solutions and error dynamics of Eq (3.5) with $\mu = 1$ via DeepONet. (a1)–(c1) Density diagrams of exact $u(x,t)$, $v(x,t)$, and $|\psi(x,t)|$. (a2)–(c2) Density diagrams of predicted $\hat{u}(x,t)$, $\hat{v}(x,t)$, and $\left|\hat{\psi}(x,t)\right|$. (a3)–(c3) Error dynamics of $|u - \hat{u}|$, $|v - \hat{v}|$, and $\left|\psi(x,t) - \hat{\psi}(x,t)\right|$.

Its solution is given by [5]

$$\psi(x,t) = \frac{A}{\sqrt{1 + a\cosh(B(x - kt))}} e^{i(kx+wt)}, \tag{3.8}$$

where $B = \sqrt{2}A$, $w = \frac{A^2}{4} - \frac{k^2}{2}$, and $a = \sqrt{1 + \frac{4}{3}\nu A^2}$.

3.3.1. PINNs method

Taking $[x_0, x_1]$ and $[t_0, t_1]$ in Eq (3.7) as $[-10, 10]$ and $[-5, 5]$, we choose the parameters and coefficients $\nu = 1$, $A = 1$, and $k = 1$. Consequently, $B = 1.4142$, $w = -0.25$, and $a = 1.52753$. The initial-boundary conditions are established using the same procedure as in Sections 3.1 and 3.2.

Unlike the previous mixed training scheme, which assigns the iterations of the Adam optimizer in advance, here we limit the Adam iterations based on the equation residual. Specifically, we first utilize the Adam optimizer to execute the program for $m_0 = 3000$ steps, and then we calculate the value of $\mathcal{R} = \frac{\mathcal{L}_{\mathcal{R}(1)} + \mathcal{L}_{\mathcal{R}(2)}}{2}$, where $\mathcal{L}_{\mathcal{R}(1)}$ and $\mathcal{L}_{\mathcal{R}(2)}$ are defined in Eq (2.7). If $\mathcal{R} \leq 0.005$, we switch to the L-BFGS optimizer for further training. If $0.005 \leq \mathcal{R} \leq 0.006$, we will add 200 more iterations of Adam training. If $\mathcal{R} > 0.006$, we will supplement 1000 iterations of Adam training. However, if the total number of iterations reaches the predefined maximum step $m = 10,000$, it will also turn to the L-BFGS optimizer. The hybrid training optimization framework, which incorporates the aforementioned modified Adam optimizer to minimize the loss function, is shown in Figure 10.

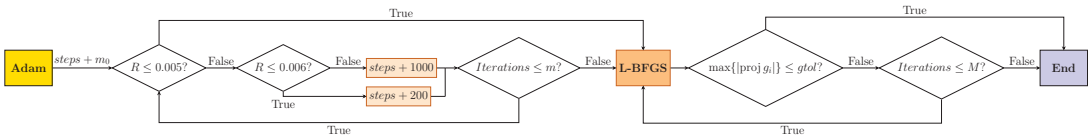


Figure 10. Mixed optimization scheme with a strategy to limit Adam iterations for minimizing the loss function.

Table 9 exhibits the numerical results obtained from solving Eq (3.7) under different optimization schemes. In addition to the widely utilized optimization techniques like SGD, Adam, L-BFGS, and the mixed optimization strategy, the approach illustrated in Figure 10 is also taken into account. Specifically, the learning rates for SGD and Adam optimization algorithms are set at 0.01 and 0.001, respectively. The mixed optimization strategy involves training with Adam for 6000 iterations, followed by switching to L-BFGS until the model converges. The findings presented in the table suggest that the SGD optimizer exhibits poor performance, achieving only an accuracy of $O(10^{-1})$. In contrast, the L-BFGS optimizer demonstrates higher accuracy than Adam. Specifically, the mixed optimization scheme achieves higher accuracy than the L-BFGS optimization, but it consumes more training time. To compare the efficiency of the two mixed optimization schemes, we observe that after 4600 steps of Adam training, the residual $\mathcal{R}$ reaches 0.005. When the same level of accuracy is targeted, the modified Adam (M-Adam) optimizer saves 1400 steps compared with the fixed-step mixed scheme. However, its total runtime increases slightly, because the additional residual checks and optimizer-switching in the first stage incur a few seconds of overhead, which is further amplified by the high-speed GPU computation. Furthermore, Table 9 reveals a strong correlation between the accuracy of the predicted solution and the PDE residual, where a smaller residual corresponds to higher accuracy.

Figure 11 displays the real and imaginary parts of the data-driven solution to Eq (3.7). The network training lasted approximately 91.72 seconds, with the optimal model achieved at the 14,789th iteration. The $\mathbb{L}^2$-norm error for the learned $\hat{u}(x, t)$ is 3.304×10^{-3} and for the learned $\hat{v}(x, t)$ is 3.342×10^{-3}. The 3D

Table 9. Comparison of various optimization strategies for addressing the NLSE (3.7) with parabolic law nonlinearity.

Schemes	$\mathbb{L}^2$-norm errors	Residual	Total iterations	Running time (s)
SGD	9.298×10^{-1}	3.222×10^{-3}	20,000	47.76
Adam	4.405×10^{-2}	7.794×10^{-4}	20,000	50.01
L-BFGS	4.988×10^{-3}	2.197×10^{-4}	7791	66.35
Adam + L-BFGS	2.472×10^{-3}	1.795×10^{-4}	6000 + 8789	91.72
M-Adam + L-BFGS	2.413×10^{-3}	1.204×10^{-4}	3000 + 1600 + 7727	95.13

visualizations of $\hat{u}(x,t)$ and $\hat{v}(x,t)$ are exhibited in Figure 11(a1),(a2), while the corresponding density diagrams are shown in Figure 11(b1),(b2). Moreover, the errors between the exact values and predictions for $\hat{u}(x,t)$ and $\hat{v}(x,t)$ are displayed in Figure 11(c1),(c2). The comparisons at $t = -3$ and $t = 3$ are presented in Figure 11(d1),(d2) and Figure 11(e1),(e2), respectively. The 3D visualization of the learned modulus $\left|\hat{\psi}(x,t)\right|$ to Eq (3.7) is exhibited in Figure 4(a3) with an $\mathbb{L}^2$-norm error of 2.472×10^{-3}. The dynamic error of $\left|\hat{\psi}(x,t)\right|$ is presented in Figure 4(b3). Furthermore, the comparisons between the exact solution and predictions at $t = -3$ and $t = 3$ are illustrated in Figure 4(c3).

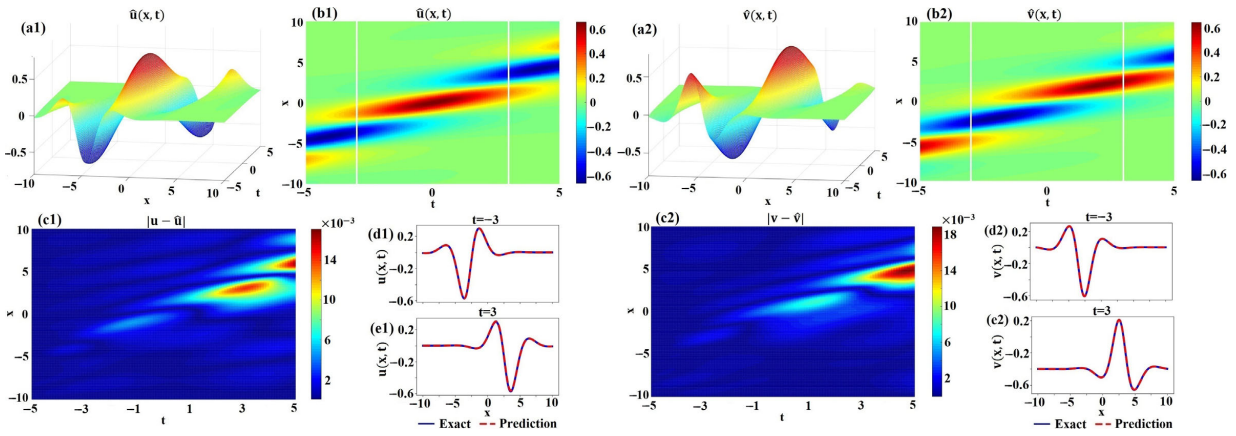


Figure 11. Data-driven real part $\hat{u}(x,t)$ and imaginary part $\hat{v}(x,t)$ of Eq (3.7) with parabolic law nonlinearity. (a1),(a2) 3D visualizations of $\hat{u}(x,t)$ and $\hat{v}(x,t)$. (b1),(b2) Density diagrams of $\hat{u}(x,t)$ and $\hat{v}(x,t)$. (c1),(c2) Error dynamics of $|u - \hat{u}|$ and $|v - \hat{v}|$. (d1),(d2) Comparisons between the exact solution and prediction at $t = -3$. (e1),(e2) Comparisons between the exact solution and prediction at $t = 3$.

3.3.2. DeepONet method

For dataset construction, we set $A = 1$ and $k = 1$ in Eq (3.8) and sample the parameter v uniformly from the interval $0.1 < v < 5$. A total of 10,000 data samples are randomly generated, with 8000 used for training and 2000 reserved for testing. The branch and trunk networks both adopt a three-layer architecture containing 128 neurons per layer, and sensor locations are fixed at $m = 50$. Training is

performed using the Adam optimizer for 100,000 steps. To study the influence of the learning rate (lr) on model convergence and stability, we compare fixed and dynamic learning rate schedules. We first evaluate four fixed learning rates: 0.0025, 0.001, 0.0005, and 0.0001. As shown in Table 10, a smaller learning rate generally results in longer training time. Among the tested values, $lr = 0.0005$ yields the best performance, with $\mathbb{L}^2$-norm errors of 6.942×10^{-3}, 6.731×10^{-3}, and 5.160×10^{-3} for $\hat{u}(x,t)$, $\hat{v}(x,t)$, and $\left|\hat{\psi}(x,t)\right|$, respectively, on the 2000 test samples. For illustration, Figure 12 compares the exact and predicted soliton solutions at five temporal snapshots for $\nu = 1$ using $lr = 0.0005$, and the corresponding density plots of $\hat{u}(x,t)$, $\hat{v}(x,t)$, and $\left|\hat{\psi}(x,t)\right|$, together with their error distributions, are detailed in Figure 13.

Table 10. Performance of Adam-optimized DeepONet for Eq (3.7) with different learning rates.

Learning rate (lr)	$\mathbb{L}^2$-norm errors (evaluated on 2000 test samples)			Final loss value	Running time (s)
	$u(x,t)$	$v(x,t)$	$\lvert\psi(x,t)\rvert$		
0.0025	1.840×10^{-1}	9.111×10^{-2}	1.198×10^{-1}	1.98×10^{-4}	367.93
0.001	2.115×10^{-2}	2.984×10^{-2}	1.847×10^{-2}	1.47×10^{-5}	368.25
0.0005	6.942×10^{-3}	6.731×10^{-3}	5.160×10^{-3}	1.21×10^{-6}	369.94
0.0001	1.085×10^{-2}	1.202×10^{-2}	8.443×10^{-3}	2.99×10^{-6}	371.95
Dynamic	5.775×10^{-3}	7.869×10^{-3}	4.265×10^{-3}	9.36×10^{-7}	402.58
Cosine Annealing	7.311×10^{-3}	5.418×10^{-3}	4.705×10^{-3}	7.87×10^{-7}	637.38

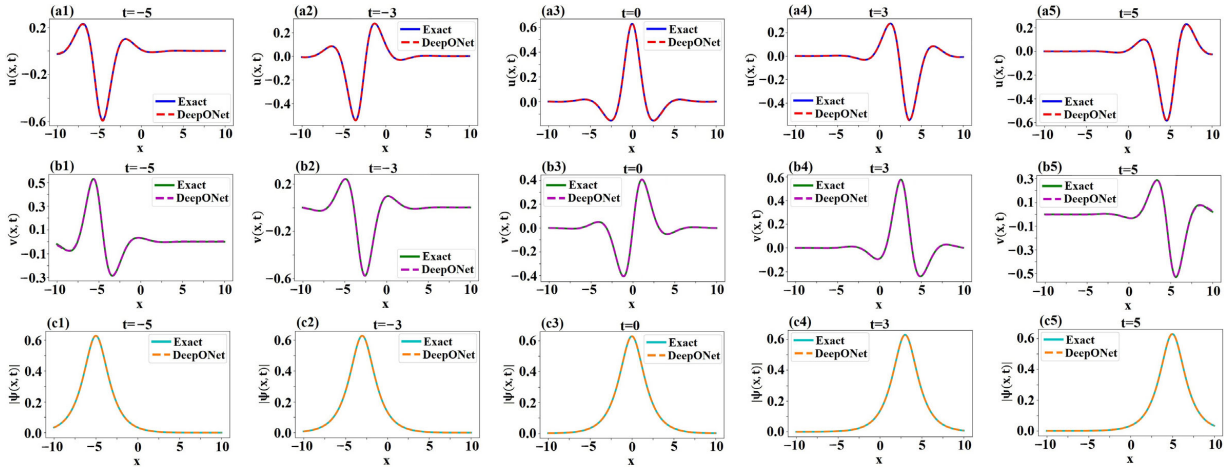


Figure 12. The results of DeepONet in solving Eq (3.7) ($\nu = 1$) with parabolic law nonlinearity. First row: Comparison between the exact and learned real part at five temporal snapshots. Second row: Comparison between the exact and learned imaginary part at five temporal snapshots. Third row: Comparison between the exact and learned modulus at five temporal snapshots.

We then examine two dynamic learning rate strategies. The first initializes lr to 0.001 and decays it by a factor of 0.9 every 10,000 training steps. The second employs cosine annealing with restarts: lr is updated every 2000 steps according to

$$lr = lr_{min} + (lr_{max} - lr_{min})\frac{1}{2}\left[1 + \cos\left(\pi \frac{k}{10,000}\right)\right], \tag{3.9}$$

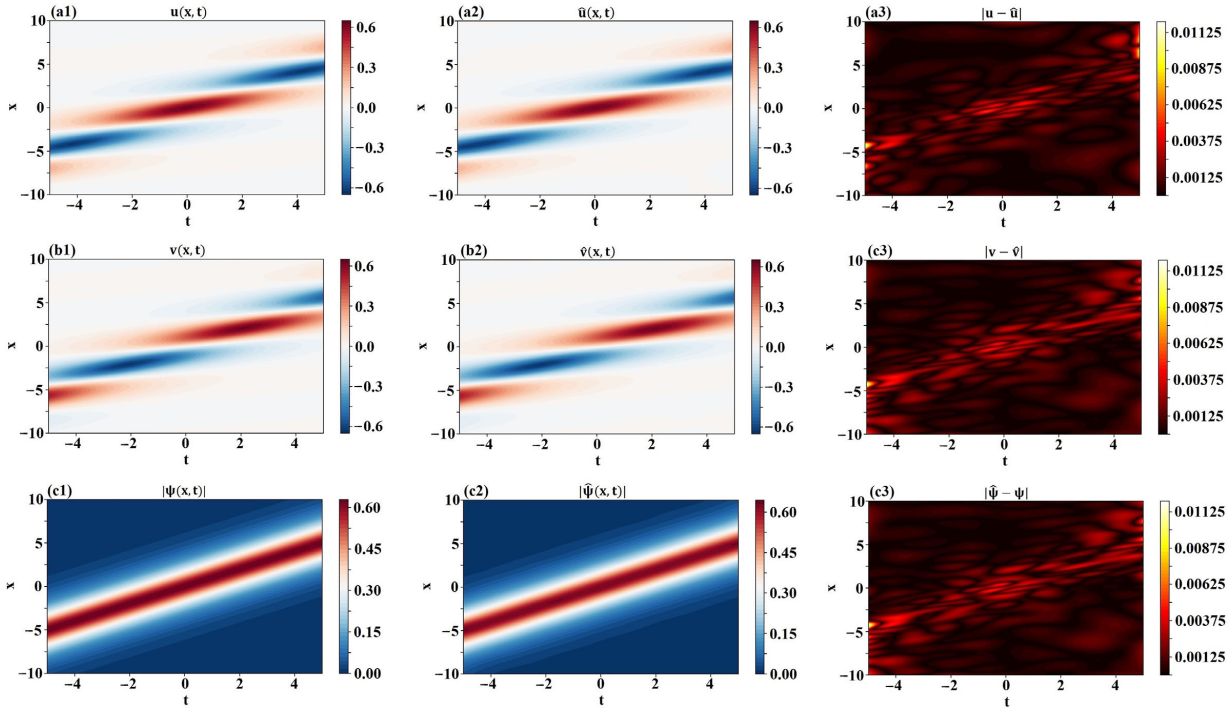


Figure 13. Data-driven solutions and error dynamics of Eq (3.7) with $\nu = 1$ via DeepONet. (a1)–(c1) Density diagrams of exact $u(x,t)$, $v(x,t)$, and $|\psi(x,t)|$. (a2)–(c2) Density diagrams of predicted $\hat{u}(x,t)$, $\hat{v}(x,t)$, and $\left|\hat{\psi}(x,t)\right|$. (a3)–(c3) Error dynamics of $|u-\hat{u}|$, $|v-\hat{v}|$, and $\left|\psi(x,t)-\hat{\psi}(x,t)\right|$.

where k denotes the number of steps elapsed since the beginning of the current cycle, each cycle consisting of 10,000 steps. Here, $lr_{max} = 0.001$ and $lr_{min} = 1 \times 10^{-5}$. The results in Table 10 demonstrate that the dynamic schedules yield more accurate predictions than any fixed learning rate. This improvement is reflected not only in the lower $\mathbb{L}^2$-norm errors but also in the reduced final loss values. However, both dynamic strategies incur additional computational cost, especially the cosine annealing.

3.4. Data-driven solution of the NLSE with logarithmic law nonlinearity

Regarding the logarithmic law nonlinearity $\mathcal{F}(|\psi|^2) = \log(|\psi|^2)$, Eq (1.1) becomes

$$i\psi_t + \frac{1}{2}\psi_{xx} + \log(|\psi|^2)\psi = 0. \tag{3.10}$$

It should be noted that the nonlinear term governing the equation is the product $\log(|\psi|^2)\psi$, rather than the logarithm alone. Since $\lim_{\psi\to 0} \log(|\psi|^2)\psi = 0$, the nonlinearity vanishes continuously at zero, and no physical quantity diverges. However, its derivative with respect to $|\psi|$ is singular at zero, which may lead to large residual gradients or ill-conditioning in gradient-based training if the predicted field visits the vacuum. In our experiments, the field amplitude did not approach zero closely enough to cause instability, owing to the specific initial-boundary data and the fact that the trained network quickly escapes the near-vacuum manifold. More generally, one may introduce the regularization $\mathcal{F}_\epsilon(|\psi|^2) = \log(|\psi|^2 + \epsilon)$ with $0 < \epsilon \ll 1$. In what follows, we will investigate it via numerical experiments.

The solution of Eq (3.10) is given by [6, 47]

$$\psi(x,t) = Ae^{-(x-kt)^2+i(kx+wt)}, \tag{3.11}$$

where $w = 2\log(A) - \frac{1}{2}k^2 - 1$. The limit $A \to 0$ corresponds to the vacuum, where the solution reduces identically to $\psi \equiv 0$. For any normalizable state with a finite particle number, A is strictly positive and w remains finite. This Gaussian-profile solution describes a chirped pulse whose width remains constant during propagation, a characteristic feature of the logarithmic nonlinearity in the absence of external potentials or higher-order effects.

3.4.1. PINNs method

The computational domain $[x_0, x_1] \times [t_0, t_1]$ for Eq (3.10) is selected as $[-10, 10]$ and $[-1, 1]$. The relevant initial-boundary conditions are derived by first selecting $A = \frac{1}{3}$, $k = 1$, and accordingly, $w = -3.69722$, and then substituting these values into Eq (3.11). Specifically, the weights assigned to each part of the loss function are set to $w_{\mathcal{R}} = 1$, $w_{\mathcal{I}} = 1$, and $w_{\mathcal{B}} = 0.01$ in this subsection. The optimization strategy (see Figure 10) proposed in Section 3.3.1 is employed to train the model.

Figure 14 displays the real and imaginary components of the data-driven solution to Eq (3.10). The network training required approximately 162.87 seconds, with the optimal model attained after 10,000 Adam iterations and 13,198 L-BFGS iterations. The $\mathbb{L}^2$-norm errors are 1.005×10^{-2} for $\hat{u}(x,t)$ and 8.601×10^{-3} for $\hat{v}(x,t)$. The 3D visualizations of $\hat{u}(x,t)$ and $\hat{v}(x,t)$ are exhibited in Figure 14(a1),(a2), while the corresponding density diagrams are displayed in Figure 14(b1),(b2). Moreover, the absolute errors between the exact values and predictions are shown in Figure 14 (c1),(c2). The comparisons of prediction accuracy at $t = -0.5$ and $t = 0.5$ are presented in Figure 14(d1),(d2) and Figure 14(e1),(e2), respectively. Furthermore, the 3D profile of the learned $\left|\hat{\psi}(x,t)\right|$ to Eq (3.10) is exhibited in Figure 4(a4) with an $\mathbb{L}^2$-norm error of 6.478×10^{-3}, while the dynamic error is presented in Figure 4(b4). The comparisons of prediction accuracy at $t = -0.6$ and $t = 0.8$ are illustrated in Figure 4(c4).

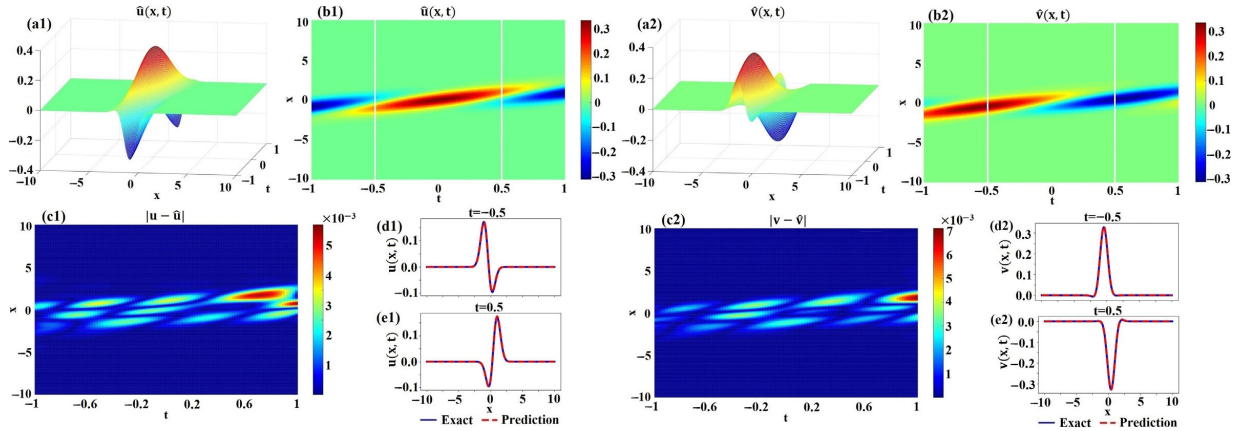


Figure 14. Data-driven real part $\hat{u}(x,t)$ and imaginary part $\hat{v}(x,t)$ of Eq (3.10) with logarithmic law nonlinearity. (a1),(a2) 3D visualizations of $\hat{u}(x,t)$ and $\hat{v}(x,t)$. (b1),(b2) Density diagrams of $\hat{u}(x,t)$ and $\hat{v}(x,t)$. (c1),(c2) Error dynamics of $|u - \hat{u}|$ and $|v - \hat{v}|$. (d1),(d2) Comparisons between the exact solution and prediction at $t = -0.5$. (e1),(e2) Comparisons between the exact solution and prediction at $t = 0.5$.

Table 11 presents a comparison of two mixed optimization schemes for solving the NLSE with Kerr, power, parabolic, and logarithmic nonlinearities. In particular, these results are obtained from ten independent experiments with different random seeds, and the mean and standard deviation are recorded. Scheme 1 employs Adam for 6000 iterations, followed by L-BFGS for model training. Scheme 2 utilizes M-Adam (see Figure 10) for initial optimization, followed by L-BFGS for fine-tuning. One advantage of Scheme 2 is that it eliminates the requirement for predetermining the number of Adam training iterations. The data present in Table 11 clearly show that when Scheme 2 is used to solve Eqs (3.1) and (3.5), the error increases slightly, while the runtime decreases. When solving Eqs (3.7) and (3.10), the error decreases significantly, especially for Eq (3.10). The experiments indicate that the dynamic adjustment of Scheme 2 increases the number of Adam training steps, thereby improving accuracy; correspondingly, the runtime also increases. Moreover, the applicability and efficiency of Scheme 2 for other nonlinear PDEs are verified in Table C1 of Appendix C. This optimization idea offers new guidance for creating more efficient optimization schemes, and future research will further refine this idea by incorporating other components of the loss function.

Table 11. The efficiency of mixed optimization Scheme 1 (6000 steps Adam + L-BFGS) and Scheme 2 (M-Adam + L-BFGS) for solving the NLSE in Kerr and non-Kerr law media.

Nonlinearity	Optimization	$\mathbb{L}^2$-norm errors	Running time (s)
Eq (3.1)	Scheme 1	$(2.34 \pm 1.14) \times 10^{-3}$	76.84 ± 7.68
	Scheme 2	$(2.89 \pm 1.30) \times 10^{-3}$	71.70 ± 5.37
Eq (3.5)	Scheme 1	$(1.01 \pm 0.25) \times 10^{-3}$	79.56 ± 5.39
	Scheme 2	$(1.78 \pm 0.36) \times 10^{-3}$	76.76 ± 9.88
Eq (3.7)	Scheme 1	$(5.12 \pm 2.39) \times 10^{-3}$	88.82 ± 9.41
	Scheme 2	$(4.12 \pm 1.88) \times 10^{-3}$	97.29 ± 12.23
Eq (3.10)	Scheme 1	$(1.17 \pm 0.51) \times 10^{-2}$	102.33 ± 21.51
	Scheme 2	$(8.02 \pm 3.73) \times 10^{-3}$	139.57 ± 13.79

Finally, we consider adding a small regularization parameter to $\mathcal{F}(|\psi|^2)$ in Eq (3.10), i.e.,

$$i\psi_t + \frac{1}{2}\psi_{xx} + \log(|\psi|^2 + \epsilon)\psi = 0, \tag{3.12}$$

where $0 < \epsilon \ll 1$. Here ϵ is treated as a numerical regularization parameter. As reported in Table 12, introducing a small ϵ yields better performance than the unregularized case ($\epsilon = 0$) in the tested experiments. However, the accuracy does not improve monotonically as ϵ decreases. This non-monotonic dependence is reported here as an empirical observation. In this study, we do not draw any conclusion about the gradient behavior of the regularized solution or about exploding gradients during training. A detailed asymptotic or optimization-theoretic analysis is beyond the scope of this work and is left for future research.

Table 12. Results of PINNs with optimization Scheme 2 for solving Eq (3.12) under different regularization parameter ϵ.

Solution	Value of ϵ					
	0.01	0.001	0.0001	0.000001	0.00000001	0
$u(x,t)$	3.234×10^{-1}	5.178×10^{-2}	9.349×10^{-3}	5.712×10^{-3}	8.463×10^{-3}	1.005×10^{-2}
$v(x,t)$	2.644×10^{-1}	4.317×10^{-2}	8.006×10^{-3}	5.145×10^{-3}	7.148×10^{-3}	8.601×10^{-3}
$\lvert\psi(x,t)\rvert$	2.142×10^{-1}	3.802×10^{-2}	6.750×10^{-3}	4.001×10^{-3}	6.861×10^{-3}	6.478×10^{-3}

3.4.2. DeepONet method

For this experiment, we select $A = \frac{1}{3}$ in Eq (3.11) and sample the parameter k from the interval $0.5 < k < 5$. All other experimental settings remain consistent with those described in Section 3.3.2. The model is trained using the Adam optimizer with a fixed learning rate of 0.0005 to minimize the loss function. In this part, we explore the impact of batch size, which is defined as the number of samples utilized for each parameter update, on model performance. According to the data in Table 13, larger batch sizes generally yield higher prediction accuracy and lower loss value at the expense of increased training time. Our analysis focuses primarily on the configuration with a batch size of 1024. After 100,000 iterations of Adam optimization, which took 368.57 seconds, the optimal model was obtained. The corresponding $\mathbb{L}^2$-norm errors evaluated on 2000 test samples are 4.334×10^{-2} for $\hat{u}(x,t)$, 5.432×10^{-2} for $\hat{v}(x,t)$, and 3.192×10^{-2} for $\left|\hat{\psi}(x,t)\right|$.

Table 13. DeepONet solutions to Eq (3.10) under varying batch sizes.

Batch size (Bs)	$\mathbb{L}^2$-norm errors (evaluated on 2000 test samples)			Final loss value	Running time (s)
	$u(x,t)$	$v(x,t)$	$\lvert\psi(x,t)\rvert$		
128	1.059×10^{-1}	1.194×10^{-1}	7.637×10^{-2}	3.22×10^{-5}	290.39
256	8.223×10^{-2}	7.560×10^{-2}	5.562×10^{-2}	1.17×10^{-5}	304.42
512	5.845×10^{-2}	7.261×10^{-2}	4.490×10^{-2}	5.25×10^{-6}	329.06
1024	4.334×10^{-2}	5.432×10^{-2}	3.192×10^{-2}	2.32×10^{-6}	368.57
2048	3.732×10^{-2}	4.508×10^{-2}	3.019×10^{-2}	1.93×10^{-6}	460.05
4096	3.947×10^{-2}	5.146×10^{-2}	3.043×10^{-2}	1.27×10^{-6}	573.46

Figure 15 displays the data-driven solutions obtained using DeepONet for Eq (3.10). Specifically, Figure 15(a1)–(a3) present comparisons of the exact and predicted real parts of the solution at $t = 0$ for $k = 1$, $k = 2.5$, and $k = 4.5$, respectively, while the corresponding imaginary parts are shown in Figure 15(b1)–(b3). The training and testing loss curves are provided in Figure 15(c). A detailed analysis for the case where $k = 1$ is presented in Figure 16. The density plots of the exact and predicted real parts are compared in Figure 16(a1),(a2), with the error distribution $|u - \hat{u}|$ visualized in Figure 16(a3). For the imaginary part, Figure 16(b1),(b2) exhibit the exact and learned results, with the error $|v - \hat{v}|$ depicted in Figure 16(b3). Additionally, the exact and predicted moduli are shown in Figure 16(c1),(c2), with the error $\left|\psi(x,t) - \hat{\psi}(x,t)\right|$ displayed in Figure 16(c3).

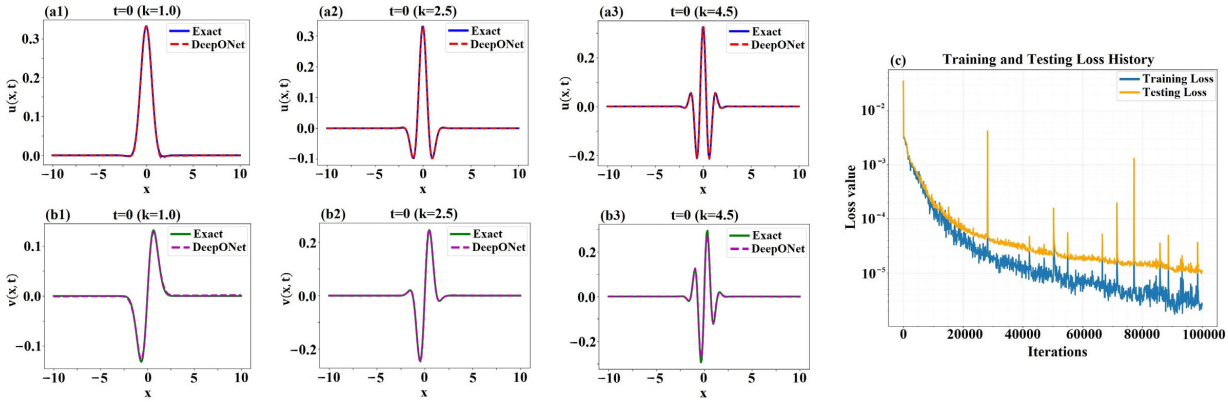


Figure 15. The results of DeepONet in solving Eq (3.10) with logarithmic law nonlinearity. (a1),(b1) $\hat{u}(x,t)$ and $\hat{v}(x,t)$ for $k = 1$ at $t = 0$. (a2),(b2) $\hat{u}(x,t)$ and $\hat{v}(x,t)$ for $k = 2.5$ at $t = 0$. (a3),(b3) $\hat{u}(x,t)$ and $\hat{v}(x,t)$ for $k = 4.5$ at $t = 0$. (c) The curve of training and testing loss.

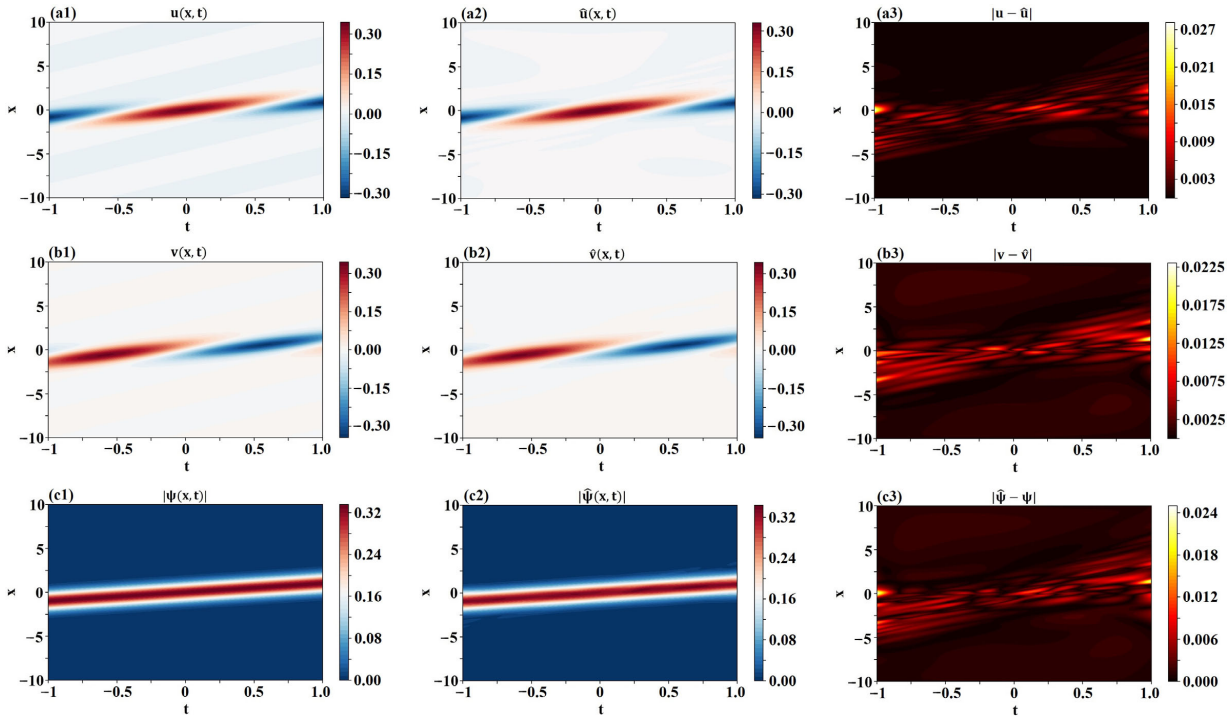


Figure 16. Data-driven solutions and error dynamics of Eq (3.10) with $k = 1$ via DeepONet. (a1)–(c1) Density diagrams of exact $u(x,t)$, $v(x,t)$, and $|\psi(x,t)|$. (a2)–(c2) Density diagrams of predicted $\hat{u}(x,t)$, $\hat{v}(x,t)$, and $\left|\hat{\psi}(x,t)\right|$. (a3)–(c3) Error dynamics of $|u - \hat{u}|$, $|v - \hat{v}|$, and $\left|\psi(x,t) - \hat{\psi}(x,t)\right|$.

4. Inverse problems

We consider the NLSE (1.1) with unknown coefficients, which can be generally expressed as

$$i\psi_t + \lambda_1\psi_{xx} + \lambda_2\mathcal{F}(|\psi|^2)\psi = 0, \tag{4.1}$$

where λ_1 and λ_2 represent the coefficients of the dispersive and nonlinear terms, respectively. As described in Section 2.1.2, a framework based on PINNs was established to identify the parameters of the NLSE in Kerr and non-Kerr law media. In this section, we employ PINNs to perform a series of numerical experiments, aiming to validate their effectiveness and robustness. Specifically, the spatio-temporal domain is $[x_0, x_1] \times [t_0, t_1]$. The selection of parameters and the initial-boundary conditions for the NLSEs under Kerr law nonlinearity and various non-Kerr law nonlinearities are consistent with those described in Section 3. A neural network comprising three hidden layers, each containing 45 neurons, was employed to address the model. The tanh function was chosen as the activation function. For training, we sampled $N_{\mathcal{R}} = 10,000$ residual points within the domain, $N_{\mathcal{I}} = 200$ points for the initial conditions, and $N_{\mathcal{B}} = 400$ points for the boundary conditions. To improve parameter identification accuracy, $N_{\mathcal{D}} = 50$ observation points were selected at a specific time $t = t'$ to provide additional information for the model. During the training process, the Adam optimizer was first employed for 10,000 steps, followed by refinement using the L-BFGS optimizer.

4.1. Discovering the coefficients of the NLSE with Kerr law nonlinearity

The NLSE in Kerr law nonlinearity with unknown coefficients λ_1 and λ_2 is given by

$$i\psi_t + \lambda_1\psi_{xx} + \lambda_2 |\psi|^2 \psi = 0. \tag{4.2}$$

In addition to the initial-boundary conditions defined in Eqs (3.3) and (3.4), observation data at the specific time $t' = 1$ are obtained from solution (3.2) as

$$\begin{cases} \mathcal{D}_u(x, 1) = \frac{1}{\cosh(x-0.5)} \cos(0.5x + 0.375), \\ \mathcal{D}_v(x, 1) = \frac{1}{\cosh(x-0.5)} \sin(0.5x + 0.375). \end{cases} \tag{4.3}$$

We adopt $N_{\mathcal{D}} = 50$ equidistant points within the interval $[-10, 10]$ at $t = 1$, and the observation information is obtained by substituting these points into Eq (4.3). The network training lasted approximately 68.43 seconds, with the optimal model concluding at the 14,861th iteration. The identified coefficients are determined as $\lambda_1 = 0.502510$ and $\lambda_2 = 0.996358$, with both relative errors confined within $O(10^{-3})$. The evolution of coefficients from the initial value 0.1 to optimized targets is illustrated in Figure 17(a).

Furthermore, Gaussian noise of varying intensities is introduced into the initial conditions to investigate the robustness of the neural network. As demonstrated in Table 14, the errors of both λ_1 and λ_2 increase with the noise level. Even with 10% initial noise, the error of the identified coefficients remains on the order of $O(10^{-2})$. In particular, we investigate the effects of low-frequency baseline drifts and high-frequency oscillations on parameter identification accuracy. From Table D3 in Appendix D, we observe that high-frequency oscillation affects λ_1 slightly more, whereas low-frequency drift has a stronger effect on λ_2. This is because high-frequency components are amplified by the second-order spatial derivative in the PDE residual, mainly affecting λ_1, while low-frequency baseline drift changes the global envelope and has a stronger effect on λ_2.

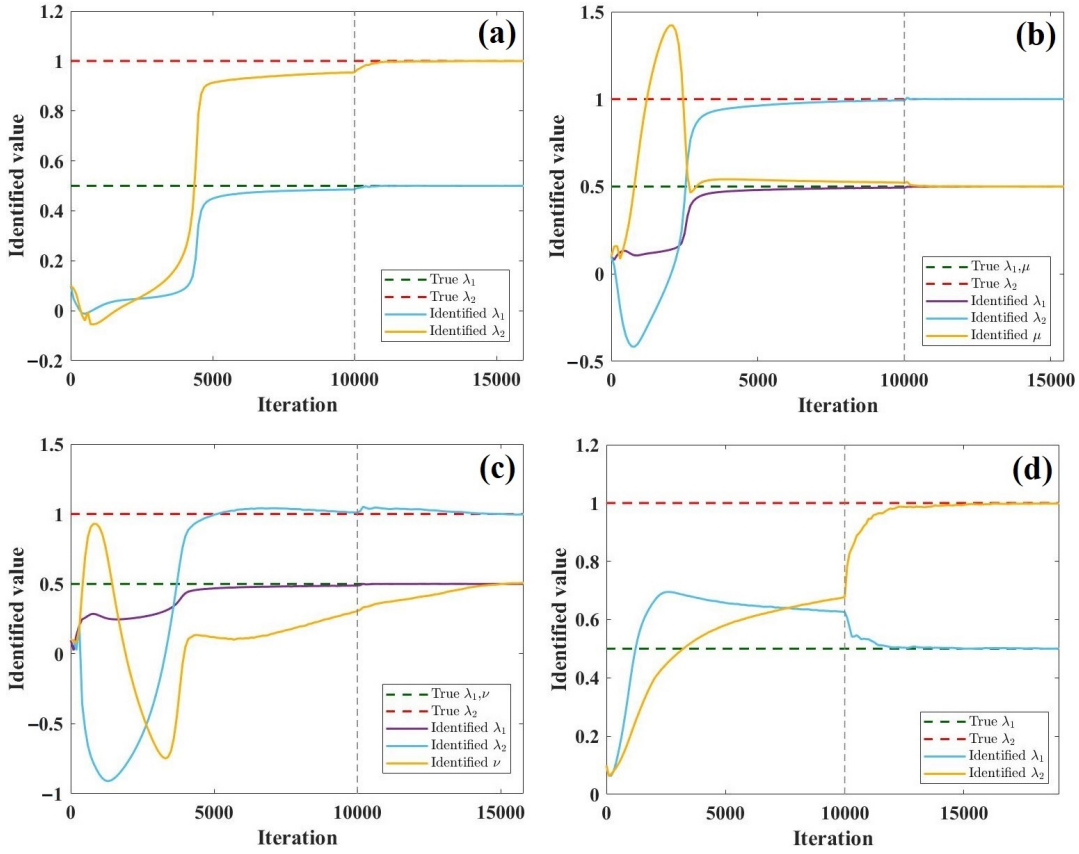


Figure 17. The process of convergence in identifying the coefficients $\boldsymbol{\lambda}$ begins with the initial value 0.1 and progresses toward the optimized targets. (a) Guided by the data derived from solutions (3.2). (b) Guided by the data derived from solutions (3.6). (c) Guided by the data derived from solutions (3.8). (d) Guided by the data derived from solutions (3.11).

Table 14. Results of identifying the coefficients $\boldsymbol{\lambda} = \{\lambda_1, \lambda_2\}$ of Eq (4.2) under varying intensities of initial Gaussian noise.

Driving data	Coefficients	Identified	Error	Total iterations	Running time (s)
Clear data	$\lambda_1 = 0.5$	0.502510	5.02×10^{-3}	14,861	68.43
	$\lambda_2 = 1$	0.996358	3.64×10^{-3}		
1% noise	$\lambda_1 = 0.5$	0.495070	9.86×10^{-3}	13,814	64.15
	$\lambda_2 = 1$	0.994922	5.08×10^{-3}		
3% noise	$\lambda_1 = 0.5$	0.509066	1.81×10^{-2}	12,935	55.30
	$\lambda_2 = 1$	0.961066	3.89×10^{-2}		
5% noise	$\lambda_1 = 0.5$	0.490394	1.92×10^{-2}	16,566	85.40
	$\lambda_2 = 1$	0.920039	7.80×10^{-2}		
10% noise	$\lambda_1 = 0.5$	0.531029	6.21×10^{-2}	25,000	150.89
	$\lambda_2 = 1$	0.903121	9.69×10^{-2}		

4.2. Discovering the coefficients of the NLSE with power law nonlinearity

The NLSE with power law nonlinearity, which contains unknown coefficients λ_1, λ_2, and a given power μ, can be expressed as

$$i\psi_t + \lambda_1\psi_{xx} + \lambda_2 |\psi|^{2\mu}\psi = 0. \tag{4.4}$$

We adopt $N_{\mathcal{D}} = 50$ equidistant points within the interval $[-10, 10]$ at time $t = 1$. The observation data, along with the initial-boundary conditions, are acquired using the same method outlined in Section 4.1. The network training required approximately 79.36 seconds, with the optimal model being attained at the 15,641th iteration. The identified coefficients are determined as $\lambda_1 = 0.4999598$, $\lambda_2 = 1.000023$, and $\mu = 0.5000382$, with all relative errors confined within $O(10^{-5})$. The evolutionary trajectories of these coefficients, from an initial value of 0.1 to their optimized targets, are illustrated in Figure 17(b). Then we examine the impact of different observation times on PINNs. The results presented in Table 15 suggest that the predictions remain stable and do not display significant fluctuations when subjected to variations in observation time. The errors associated with the identified λ_1 and λ_2 range from $O(10^{-4})$ to $O(10^{-5})$, while the error for the identified parameter μ falls within the range of $O(10^{-3})$ to $O(10^{-5})$.

Table 15. Results of identifying the coefficients $\boldsymbol{\lambda} = \{\lambda_1, \lambda_2, \mu\}$ of Eq (4.4) obtained at various observation moments t'.

Moments	Errors			Total iterations	Running time (s)
	$\lambda_1 = 0.5$	$\lambda_2 = 1$	$\mu = 0.5$		
$t' = -2$	8.59×10^{-4}	7.88×10^{-4}	7.52×10^{-4}	15,341	76.58
$t' = -1$	8.06×10^{-5}	1.04×10^{-4}	1.14×10^{-3}	16,494	84.35
$t' = 0$	6.04×10^{-5}	1.01×10^{-4}	6.10×10^{-5}	17,282	91.81
$t' = 1$	8.04×10^{-5}	2.30×10^{-5}	7.64×10^{-5}	15,641	79.36
$t' = 1.5$	4.52×10^{-5}	2.32×10^{-5}	7.60×10^{-4}	15,943	79.72

4.3. Discovering the coefficients of the NLSE with parabolic law nonlinearity

The NLSE with parabolic law nonlinearity and unknown parameters λ_1, λ_2, and ν is given by

$$i\psi_t + \lambda_1\psi_{xx} + \lambda_2\left(|\psi|^2 + \nu|\psi|^4\right)\psi = 0. \tag{4.5}$$

We adopt $N_{\mathcal{D}} = 50$ equidistant points within the interval $[-10, 10]$ at $t = 2$, while maintaining other experimental conditions consistent with the case described in Section 4.2. Specifically, we provide an initial estimate for the unknown parameters, setting $\lambda_1 = \lambda_2 = \nu = 0.1$. When using the data from NLSE (3.7) with parabolic law nonlinearity to identify the parameters of Eq (4.5), the training of the network required approximately 100.70 seconds, and the optimal model was achieved at the 18,331th iteration. The predicted values are $\lambda_1 = 0.500073$, $\lambda_2 = 1.014093$, and $\nu = 0.947352$. The relative errors are between $O(10^{-2})$ and $O(10^{-4})$. Moreover, the convergence process of the identified parameters is collected and illustrated in Figure 17(c). Table 16 provides a comparative analysis of the identified parameters obtained by varying selections of the term ν in Eq (4.5). The results presented in Table 16 suggest that the parameter ν is more difficult to estimate than the coefficients λ_1 and λ_2, with smaller values of ν being particularly challenging.

Table 16. Results of identifying the coefficients $\boldsymbol{\lambda} = \{\lambda_1, \lambda_2, \nu\}$ of Eq (4.5) based on the selection of various parameter values for ν within the driven data.

Coefficients	Identified	Error	Total iterations	Running time (s)
$\lambda_1 = 0.5$	0.499978	4.40×10^{-5}		
$\lambda_2 = 1$	1.012143	1.21×10^{-2}	16,905	87.22
$\nu = 0.25$	0.215589	1.38×10^{-1}		
$\lambda_1 = 0.5$	0.499891	2.18×10^{-4}		
$\lambda_2 = 1$	0.993724	6.28×10^{-3}	16,604	85.94
$\nu = 0.5$	0.520095	4.02×10^{-2}		
$\lambda_1 = 0.5$	0.500151	3.02×10^{-4}		
$\lambda_2 = 1$	1.003236	3.24×10^{-3}	16,488	86.44
$\nu = 0.75$	0.741055	1.19×10^{-2}		
$\lambda_1 = 0.5$	0.500073	1.46×10^{-4}		
$\lambda_2 = 1$	1.014093	1.41×10^{-2}	18,331	100.70
$\nu = 1$	0.947352	5.26×10^{-2}		

4.4. Discovering the coefficients of the NLSE with logarithmic law nonlinearity

The NLSE with logarithmic law nonlinearity and unknown coefficients λ_1 and λ_2 is formulated as

$$i\psi_t + \lambda_1\psi_{xx} + \lambda_2 \log(|\psi|^2)\psi = 0. \tag{4.6}$$

We utilize $N_{\mathcal{D}} = 50$ equidistant points in the interval $[-10, 10]$ at $t = -0.2$. The experimental settings are consistent with those outlined in Section 4.2. Using data from NLSE (3.10) to identify the parameters of Eq (4.6), the training process for the network required approximately 98.97 seconds, with the optimal model ending in the 18,305th iteration. The predicted values are $\lambda_1 = 0.5009037$ and $\lambda_2 = 0.9988502$, with both relative errors being $O(10^{-3})$. Furthermore, the convergence process of the identified parameters is collected and illustrated in Figure 17(d). The parameter evolution curves in Figure 17 exhibit pronounced oscillations during the early stage of Adam training. This behavior arises primarily from the parameters, which introduce stiffness and competing gradients, as reported in [48]. Exploring strategies to mitigate this issue, such as decoupled optimization (first training the network weights and then refining the physical parameters) and embedding physical parameters directly into the network architecture (e.g., as inputs or learnable activation layers) remains a meaningful research direction.

The errors in the estimation of coefficients for the NLSE with logarithmic law nonlinearity, corresponding to various observation points N_O, are presented in Table 17. These data reveal that the errors associated with the identified coefficients range from $O(10^{-4})$ to $O(10^{-3})$, thereby underscoring another advantage of PINNs in solving inverse problems for the NLSE with limited observational data. Furthermore, using the same dataset and the 50 supervised points described above, we identify the parameter ϵ in Eq (3.12). Since the data are generated from Eq (3.10), the parameter ϵ should theoretically approach zero in this inverse problem. After training, the identified ϵ is 5.183662×10^{-5} on clean data, and it rises to 1.560087×10^{-2} when 10% noise is added. These findings further highlight the potential of PINNs in discovering the small parameter.

Table 17. Results of discovering the coefficients $\boldsymbol{\lambda} = \{\lambda_1, \lambda_2\}$ of Eq (4.6) using varying numbers of observation points.

Points	Errors		Total iterations	Running time (s)
	$\lambda_1 = 0.5$	$\lambda_2 = 1$		
10	6.61×10^{-3}	6.75×10^{-3}	18,777	100.18
20	2.34×10^{-4}	7.02×10^{-4}	18,382	95.67
50	1.81×10^{-3}	1.15×10^{-3}	18,305	98.97
80	3.92×10^{-4}	2.13×10^{-3}	17,756	89.07
100	8.86×10^{-4}	4.29×10^{-4}	19,637	109.23

5. Summary and discussions

In this paper, we focus on deep learning methods for nonlinear Schrödinger equations (NLSEs) with Kerr and non-Kerr law nonlinearities. This is achieved by employing physics-informed neural networks (PINNs) and the deep operator network (DeepONet) to reconstruct the optical soliton solutions, while PINNs are further applied to tackle the inverse problems of parameter identification. Some key factors that influence the performance of these models are systematically explored, from which we draw conclusions regarding effective network architectures and training methodologies.

In forward problems, we successfully simulate and analyze the real part, imaginary part, and modulus of optical soliton solutions for the NLSE across various nonlinearities. For Kerr law nonlinearity, PINNs achieve a prediction accuracy of $O(10^{-3})$, whereas DeepONet attains an average accuracy of $O(10^{-2})$ over 2000 test samples. We also examine the influence of network architectures, particularly the complexity of the branch and trunk networks in DeepONet, on model performance. For power law nonlinearity, PINNs yield an accuracy between $O(10^{-4})$ and $O(10^{-3})$ by adjusting the weights of loss components. Besides, DeepONet reaches $O(10^{-3})$ accuracy when the size of the training dataset and the number of sensor locations are optimized. For parabolic law nonlinearity, we design an optimization scheme for PINNs that dynamically controls Adam iterations based on PDE residuals and achieves an accuracy of $O(10^{-3})$. The applicability and efficiency of the proposed scheme are further verified on different NPDEs (see Tables 11 and C1). In DeepONet, we investigate the effect of fixed versus dynamic learning rate schedules (e.g., the decay strategy and cosine annealing) on convergence and stability, also yielding an accuracy of $O(10^{-3})$. For logarithmic law nonlinearity, we simulate the solution under $O(10^{-3})$ to $O(10^{-2})$ accuracy. Additionally, the impact of batch size on DeepONet is explored, with the model achieving an accuracy of $O(10^{-2})$. For inverse problems, we not only identify key equation parameters such as dispersive and nonlinear coefficients as well as power exponents, but also investigate the impact of noise interference, parameter selection, observation time instants, and points on the inversion accuracy. Numerical experiments demonstrate that with the introduction of 10% Gaussian noise to the initial conditions, the relative error of the identified coefficients can reach $O(10^{-2})$. Furthermore, the identified values exhibit minimal fluctuations in response to changes in observation time and observation points. Particularly, the effects of low-frequency baseline drifts and high-frequency oscillations in the initial conditions on solution reconstruction accuracy and parameter identification accuracy of the NLSE are also analyzed (see Appendix D).

In summary, PINNs exhibit powerful capability in solving both forward and inverse problems of NLSEs across Kerr-type and non-Kerr media. While PINNs generally achieve higher prediction accuracy in simulating optical soliton solutions, DeepONet offers the distinct advantage of providing real-time predictions for scenarios with previously unseen parameters once trained. These results offer new insights into model extension and algorithm improvement. Despite these promising findings, several limitations and open questions remain for further investigation. A key limitation of the present study is the reliance on purely data-driven DeepONet. In fact, DeepONet does not require closed-form analytical solutions, as training data can be produced by high-accuracy numerical solvers. Our further work will aim to enhance this framework to mitigate the reliance on labeled data and extend it to more general forms of Eq (1.1), including cases with variable coefficients, perturbations, higher-order dispersion, nonlocal terms, fractional derivatives, $\mathcal{PT}$-symmetric potentials, and complex boundary conditions. Incorporating multi-soliton, breather, and rogue wave solutions into operator learning frameworks is also an important direction for future studies. The scope of operator learning in this study is confined to mapping from the initial condition space to the solution space. Future research will extend this framework to learn general operators, such as $\mathcal{F}(\cdot)$ in Eq (1.1), which will require the design of more sophisticated numerical schemes and function spaces. Moreover, DeepONet has been adopted across several paradigms for solving inverse problems, including serving as a forward surrogate to accelerate Bayesian inference, learning inverse operators directly from observation, performing physics-informed parameter identification, and constructing hybrid frameworks coupled with evolutionary or generative models. A systematic study of DeepONet-based inverse problems under both noise-free and noisy settings remains a promising direction.

Use of AI tools declaration

The authors declare they have not used Artificial Intelligence (AI) tools in the creation of this article.

Acknowledgments

The work was funded by the National Natural Science Foundation of China (No. 12172199).

Conflict of interest

Manwai Yuen and Lijun Zhang are guest editors for the special issue of the ERA and was not involved in the editorial review or the decision to publish this article. All authors declare that there are no competing interests.

References

1. T. Ozawa, S. F. Tian, Z. J. Wu, Orbital stability of dark solitons for the modified Korteweg-de Vries hierarchy with nonzero boundary conditions, *Math. Ann.*, **394** (2026), 23. https://doi.org/10.1007/s00208-026-03316-9
2. J. J. Yang, S. F. Tian, Z. Q. Li, The modified Camassa-Holm equation with nonzero background: Soliton resolution conjecture and asymptotic stability of N-soliton solutions, *Adv. Math.*, **484** (2026), 110552. https://doi.org/10.1016/j.aim.2025.110552

3. Y. S. Kivshar, G. P. Agrawal, *Optical Solitons: From Fibers to Photonic Crystals*, Academic Press, New York, 2003.
4. G. P. Agrawal, *Nonlinear Fiber Optics*, Academic Press, San Diego, 2019.
5. R. Kohl, A. Biswas, D. Milovic, E. Zerrad, Optical soliton perturbation in a non-Kerr law media, *Opt. Laser Technol.*, **40** (2008), 647–662. https://doi.org/10.1016/j.optlastec.2007.10.002
6. A. Biswas, S. Konar, *Introduction to Non-Kerr Law Optical Solitons*, CRC Press, New York, 2006.
7. M. J. Ablowitz, H. Segur, *Solitons and the Inverse Scattering Transform*, SIAM, Philadelphia, 1981.
8. J. Yang, *Nonlinear Waves in Integrable and Nonintegrable Systems*, SIAM, Philadelphia, 2010.
9. N. Akhmediev, A. Ankiewicz, J. M. Soto-Crespo, Rogue waves and rational solutions of the nonlinear Schrödinger equation, *Phys. Rev. E*, **80** (2009), 026601. https://doi.org/10.1103/PhysRevE.80.026601
10. L. Gagnon, P. Winternitz, Lie symmetries of a generalised nonlinear Schrödinger equation: I. The symmetry group and its subgroups, *J. Phys. A: Math. Gen.*, **21** (1988), 1493. https://doi.org/10.1088/0305-4470/21/7/013
11. M. Wadati, H. Sanuki, K. Konno, Relationships among inverse method, Bäcklund transformation and an infinite number of conservation laws, *Prog. Theor. Phys.*, **53** (1975), 419–436. https://doi.org/10.1143/PTP.53.419
12. P. D. Lax, Integrals of nonlinear equations of evolution and solitary waves, *Commun. Pure Appl. Math.*, **21** (1968), 467–490. https://doi.org/10.1002/cpa.3160210503
13. R. Hirota, *The Direct Method in Soliton Theory*, Cambridge University Press, Cambridge, 2004. https://doi.org/10.1017/CBO9780511543043
14. M. J. Ablowitz, P. A. Clarkson, *Solitons, Nonlinear Evolution Equations and Inverse Scattering*, Cambridge University Press, Cambridge, 1991. https://doi.org/10.1017/CBO9780511623998
15. A. D. Polyanin, N. A. Kudryashov, Closed-form solutions of the nonlinear Schrödinger equation with arbitrary dispersion and potential, *Chaos, Solitons Fractals*, **191** (2025), 115822. https://doi.org/10.1016/j.chaos.2024.115822
16. S. Ibrahim, D. Baleanu, Classes of solitary solution for nonlinear Schrödinger equation arising in optical fibers and their stability analysis, *Opt. Quantum Electron.*, **55** (2023), 1158. https://doi.org/10.1007/s11082-023-05423-2
17. L. Barletti, L. Brugnano, G. Gurioli, F. Iavernaro, Recent advances in the numerical solution of the nonlinear Schrödinger equation, *J. Comput. Appl. Math.*, **445** (2024), 115826. https://doi.org/10.1016/j.cam.2024.115826
18. S. Huang, W. Feng, C. Tang, Z. He, C. Yu, J. Lv, Partial differential equations meet deep neural networks: a survey, *IEEE Trans. Neural Networks Learn. Syst.*, **36** (2025), 13649–13669. https://doi.org/10.1109/TNNLS.2025.3545967
19. M. Raissi, P. Perdikaris, G. E. Karniadakis, Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, *J. Comput. Phys.*, **378** (2019), 686–707. https://doi.org/10.1016/j.jcp.2018.10.045

20. A. D. Jagtap, G. E. Karniadakis, Extended physics-informed neural networks (XPINNs): A generalized space-time domain decomposition based deep learning framework for nonlinear partial differential equations, *Commun. Comput. Phys.*, **28** (2020), 2002–2041. https://doi.org/10.4208/cicp.OA-2020-0164

21. J. Yu, L. Lu, X. H. Meng, G. E. Karniadakis, Gradient-enhanced physics-informed neural networks for forward and inverse PDE problems, *Comput. Methods Appl. Mech. Eng.*, **393** (2022), 114823. https://doi.org/10.1016/j.cma.2022.114823

22. X. Y. Yang, Z. Wang, Solving Benjamin-Ono equation via gradient balanced PINNs approach, *Eur. Phys. J. Plus*, **137** (2022), 864. https://doi.org/10.1140/epjp/s13360-022-03078-8

23. C. Wu, M. Zhu, Q. Tan, Y. Kartha, L. Lu, A comprehensive study of non-adaptive and residual-based adaptive sampling for physics-informed neural networks, *Comput. Methods Appl. Mech. Eng.*, **403** (2023), 115671. https://doi.org/10.1016/j.cma.2022.115671

24. A. D. Jagtap, E. Kawaguchi, G. E. Karniadakis, Adaptive activation functions accelerate convergence in deep and physics-informed neural networks, *J. Comput. Phys.*, **404** (2020), 109136. https://doi.org/10.1016/j.jcp.2019.109136

25. J. D. Toscano, V. Oommen, A. J. Varghese, Z. Zou, N. A. Daryakenari, C. Wu, et al., From PINNs to PIKANs: recent advances in physics-informed machine learning, *Mach. Learn. Comput. Sci. Eng.*, **1** (2025), 15. https://doi.org/10.1007/s44379-025-00015-1

26. L. Lu, R. Pestourie, W. Yao, Z. Wan, F. Verdugo, S. G. Johnson, Physics-informed neural networks with hard constraints for inverse design, *SIAM J. Sci. Comput.*, **43** (2021), B1105–B1132. https://doi.org/10.1137/21M1397908

27. L. Lu, P. Z. Jin, G. F. Pang, Z. Q. Zhang, G. E. Karniadakis, Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators, *Nat. Mach. Intell.*, **3** (2021), 218–229. https://doi.org/10.1038/s42256-021-00302-5

28. T. Chen, H. Chen, Universal approximation to nonlinear operators by neural networks with arbitrary activation functions and its application to dynamical systems, *IEEE Trans. Neural Networks*, **6** (1995), 911–917. https://doi.org/10.1109/72.392253

29. M. Zhu, H. Zhang, A. Jiao, G. E. Karniadakis, L. Lu, Reliable extrapolation of deep neural operators informed by physics or sparse observations, *Comput. Methods Appl. Mech. Eng.*, **412** (2023), 116064. https://doi.org/10.1016/j.cma.2023.116064

30. S. F. Wang, H. W. Wang, P. Perdikaris, Learning the solution operator of parametric partial differential equations with physics-informed DeepONets, *Sci. Adv.*, **7** (2021), eabi8605. https://doi.org/10.1126/sciadv.abi8605

31. Y. H. Yang, Deeponet for solving nonlinear partial differential equations with physics-informed training, *Neural Networks*, **197** (2025), 108490. https://doi.org/10.1016/j.neunet.2025.108490

32. J. C. Pu, J. Li, Y. Chen, Soliton, breather, and rogue wave solutions for solving the nonlinear Schrödinger equation using a deep learning method with physical constraints, *Chin. Phys. B*, **30** (2021), 060202. https://doi.org/10.1088/1674-1056/abd7e3

33. J. C. Pu, J. Li, Y. Chen, Solving localized wave solutions of the derivative nonlinear Schrödinger equation using an improved PINN method, *Nonlinear Dyn.*, **105** (2021), 1723–1739. https://doi.org/10.1007/s11071-021-06554-5

34. W. B. Bo, R. R. Wang, Y. Fang, Y. Y. Wang, C. Q. Dai, Prediction and dynamical evolution of multipole soliton families in fractional Schrödinger equation with the $\mathcal{PT}$-symmetric potential and saturable nonlinearity, *Nonlinear Dyn.*, **111** (2023), 1577–1588. https://doi.org/10.1007/s11071-022-07884-8

35. Z. J. Zhou, Z. Y. Yan, Solving forward and inverse problems of the logarithmic nonlinear Schrödinger equation with $\mathcal{PT}$-symmetric harmonic potential via deep learning, *Phys. Lett. A*, **387** (2021), 127010. https://doi.org/10.1016/j.physleta.2020.127010

36. J. Song, Z. Y. Yan, Deep learning soliton dynamics and complex potentials recognition for 1D and 2D $\mathcal{PT}$-symmetric saturable nonlinear Schrödinger equations, *Physica D*, **448** (2023), 133729. https://doi.org/10.1016/j.physd.2023.133729

37. Y. R. Zhang, F. J. Liu, Noether-constrained physics-informed neural networks for the nonlinear Schrödinger equation, *Phys. Lett. A*, **589** (2026), 131802. https://doi.org/10.1016/j.physleta.2026.131802

38. Y. Zhang, X. Lv, Data-driven solutions and parameter discovery of the extended higher-order nonlinear Schrödinger equation in optical fibers, *Physica D*, **468** (2024), 134284. https://doi.org/10.1016/j.physd.2024.134284

39. X. Peng, Y. W. Zhao, X. Lv, Data-driven solitons and parameter discovery to the (2+1)-dimensional NLSE in optical fiber communications, *Nonlinear Dyn.*, **112** (2024), 1291–1306. https://doi.org/10.1007/s11071-023-09083-5

40. J. H. Wang, X. Wang, J. Li, B. Liu, Data generation-based operator learning for solving partial differential equations on unbounded domains, *Commun. Comput. Phys.*, **37** (2025), 1383–1416. https://doi.org/10.4208/cicp.OA-2023-0327

41. J. Z. Huang, R. K. You, T. Zhou, Deep learning for the semi-classical limit of the Schrödinger equation, *J. Comput. Phys.*, **558** (2026), 114869. https://doi.org/10.1016/j.jcp.2026.114869

42. Q. Ma, X. Jiang, X. Guo, Q. Zhou, X. Wang, H. Yu, et al., Layer-interaction DeepONet for modeling ultrafast nonlinear dynamics in optical fibers, *J. Lightwave Technol.*, **43** (2025), 9403–9413. https://doi.org/10.1109/JLT.2025.3598027

43. D. P. Kingma, J. Ba, Adam: a method for stochastic optimization, preprint, arXiv:1412.6980. https://doi.org/10.48550/arXiv.1412.6980

44. D. Liu, J. Nocedal, On the limited memory BFGS method for large scale optimization, *Math. Program.*, **45** (1989), 503–528. https://doi.org/10.1007/BF01589116

45. X. L. Wang, Z. K. Wu, W. J. Han, Z. Y. Yan, Deep learning data-driven multi-soliton dynamics and parameters discovery for the fifth-order Kaup-Kuperschmidt equation, *Physica D*, **454** (2023), 133862. https://doi.org/10.1016/j.physd.2023.133862

46. A. M. Kamchatnov, New approach to periodic solutions of integrable equations and nonlinear theory of modulational instability, *Phys. Rep.*, **286** (1997), 199–270. https://doi.org/10.1016/S0370-1573(96)00049-X

47. L. Girgis, D. Milovic, T. Hayat, O. M. Aldossary, A. Biswas, Optical soliton perturbation with log law nonlinearity, *Opt. Appl.*, **42** (2012), 447–454. https://doi.org/10.5277/oa120301

48. S. F. Wang, Y. J. Teng, P. Perdikaris, Understanding and mitigating gradient flow pathologies in physics-informed neural networks, *SIAM J. Sci. Comput.*, **43** (2021), A3055–A3081. https://doi.org/10.1137/20M1318043

49. A. M. Wazwaz, *Partial Differential Equations and Solitary Waves Theory*, Springer Berlin, Heidelberg, 2009. https://doi.org/10.1007/978-3-642-00251-9

50. A. M. Wazwaz, Multiple-soliton solutions for the KP equation by Hirota's bilinear method and by the tanh-coth methoh, *Appl. Math. Comput.*, **190** (2007), 633–640. https://doi.org/10.1016/j.amc.2007.01.056

51. S. C. Anco, N. T. Ngatat, M. Willoughby, Interaction properties of complex modified Korteweg-de Vries (mKdV) solitons, *Physica D*, **240** (2011), 1378–1394. https://doi.org/10.1016/j.physd.2011.06.003

52. Y. Zhang, L. Guo, J. He, Z. Zhou, Darboux transformation of the second-type derivative nonlinear Schrödinger equation, *Lett. Math. Phys.*, **105** (2015), 853–891. https://doi.org/10.1007/s11005-015-0758-x

53. P. L. Ma, J. C. Pu, W. Q. Peng, PINN for solving localized wave solutions and inverse problems involving the Gerdjikov-Ivanov equation, *Nonlinear Dyn.*, **113** (2025), 26547–26559. https://doi.org/10.1007/s11071-025-11457-w

Appendix

A. Summary of investigated factors and configurations

Table A1 provides a quick reference for the factors examined in different parts of this study.

Table A1. Summary of factors affecting model performance examined in this study.

	Nonlinearity	Method	Factors studied	Configuration adopted
Section 3.1.1		PINNs	Network architecture	$[45] \times 3$
Section 3.1.2	Kerr	DeepONet	Network architecture (branch & trunk)	$[128] \times 3$
Section 4.1		PINNs (Inverse)	Low- and high-frequency noise interference	0–10%
Section 3.2.1		PINNs	Weight	$w_{\mathcal{R}} = 1$, $w_{\mathcal{I}} = 0.1$, $w_{\mathcal{B}} = 0.01$
Section 3.2.2	Power	DeepONet	Data size & Sensor configuration	$S = 10,000$, $m = 50$
Section 4.2		PINNs (Inverse)	Observation time	$t' = 1$
Section 3.3.1		PINNs	Optimization scheme	M-Adam & L-BFGS
Section 3.3.2	Parabolic	DeepONet	Learning rate	$lr = 0.0005$
Section 4.3		PINNs (Inverse)	Unknown parameter	$\nu \in \{0.25, 0.5, 0.75, 1\}$
Section 3.4.1		PINNs	Regularization parameter	Smaller ϵ
Section 3.4.2	Logarithmic	DeepONet	Batch size	$Bs = 1024$
Section 4.4		PINNs (Inverse)	Observation point	$N_{\mathcal{D}} = 50$

B. Supplementary results of DeepONet

This section provides supplementary quantitative results for DeepONet, including its online inference efficiency, extrapolation capability, and sensitivity to changes in sensor locations. In addition, the numbers of network parameters in PINNs and DeepONet are reported.

B.1. Inference efficiency

The real-time inference efficiency of DeepONet in predicting optical solitons is investigated using the standard NLSE (3.1) as an example. Specifically, the network architecture and hyperparameters are consistent with those described in Section 3.1.2. We evaluate the average inference time for a single initial condition in two scenarios: single-time prediction and full spatiotemporal field prediction. In the first scenario, DeepONet learns the mapping $\psi(x, t_0) \mapsto \hat{\psi}(x^*, t')$, and the prediction at t' is tested at 200 spatial points. In the second scenario, DeepONet learns the mapping $\psi(x, t_0) \mapsto \hat{\psi}(x^*, t^*)$, and the prediction is evaluated at selected spatiotemporal points over the full domain. As shown in Table B1, each scenario is evaluated on 50 different initial conditions. For single-time profile prediction with a batch size of 200 points, DeepONet can process 650–700 initial conditions per second. For spatiotemporal evolution prediction, the average inference time increases with the batch size. On average, a throughput of 350–400 predictions per second is achieved with a batch size of 10,000 points, at which point the computational capacity is nearly saturated.

Table B1. Real-time inference efficiency of DeepONet in predicting the optical solitons of the standard NLSE (3.1).

Orientation	Deployment	Batch size	Error of $\lvert\hat{\psi}(x,t)\rvert$	Time (milliseconds)	Throughput (per second)
Single-time profile prediction ($t' = 0$)	$\psi(x, t_0) \mapsto \hat{\psi}(x^*, t')$	200	3.346×10^{-2}	1.52	659
Single-time profile prediction ($t' = 5$)	$\psi(x, t_0) \mapsto \hat{\psi}(x^*, t')$	200	6.909×10^{-2}	1.45	688
Spatiotemporal evolution prediction	$\psi(x, t_0) \mapsto \hat{\psi}(x^*, t^*)$	200×100	2.849×10^{-2}	5.93	169
Spatiotemporal evolution prediction	$\psi(x, t_0) \mapsto \hat{\psi}(x^*, t^*)$	400×200	2.934×10^{-2}	19.65	51

B.2. Extrapolation capability

In Section 3.1.2, we assessed the interpolation capability of DeepONet for constructing optical solitons of the standard NLSE (3.1) within the training range, achieving an accuracy of $O(10^{-2})$. This subsection is dedicated to evaluating the extrapolation capability of DeepONet beyond the training ranges of the input parameter A and time t. On the one hand, we trained the model for $A \in [0.1, 2]$ and sampled 100 test initial conditions with different values of A in unseen ranges. The extrapolation error was evaluated over the entire domain using a 200×100 spatiotemporal grid. As shown in Table B2, for unseen A, the prediction error is on the order of $O(10^{-1})$. On the other hand, we trained the model for $t \in [-5, 5]$ and tested it outside this range. Specifically, we adopted 100 time instants within the new ranges and assessed the accuracy using 200 spatial points for each instant. For unseen t, the prediction error is also on the order of $O(10^{-1})$, which is smaller than that for unseen A, indicating better extrapolation performance in time. However, the extrapolation error is approximately one order of magnitude larger than the interpolation error.

Table B2. Extrapolation capability of DeepONet in predicting the optical solitons of the standard NLSE (3.1).

Unseen range of A			Unseen range of t		
$[2, 2.25]$	$[2, 2.5]$	$[2, 3]$	$[5, 5.5]$	$[5, 6]$	$[-6, -5]$
$0.3793 \pm 8.66\%$	$0.3289 \pm 8.73\%$	$0.4733 \pm 11.74\%$	$0.1202 \pm 2.90\%$	$0.1471 \pm 5.34\%$	$0.1612 \pm 5.96\%$

B.3. Sensitivity to changes in sensor locations

In Section 3.2.2, we explored the influence of sensor numbers m on the prediction accuracy of Eq (3.5) with power law nonlinearity. In that analysis, however, the sensor locations were fixed and uniformly spaced. This section examines the sensitivity of DeepONet to the distribution of initial-condition sensor locations by independent uniform random sampling and Latin hypercube sampling (LHS). The dataset size is fixed at 10,000, and all other parameters are the same as those described in Section 3.2.2. The results in Table B3 indicate that the prediction results are insensitive to the sampling strategy of the initial-condition sensors.

Table B3. Sensitivity of DeepONet to the distribution of initial-condition sensor locations.

	10	20	50	100	200	300
Equidistant	7.665×10^{-3}	9.742×10^{-3}	6.156×10^{-3}	1.306×10^{-2}	1.467×10^{-2}	2.369×10^{-2}
Random	8.353×10^{-3}	4.705×10^{-3}	9.799×10^{-3}	1.380×10^{-2}	2.309×10^{-2}	1.621×10^{-2}
LHS	7.952×10^{-3}	9.629×10^{-3}	9.415×10^{-3}	1.713×10^{-2}	1.519×10^{-2}	1.695×10^{-2}

B.4. Number of network parameters of DeepONet and PINNs

Table B4 lists the numbers of network parameters for PINNs and DeepONet in our study. PINNs use only 4367 trainable parameters, including 4230 weights and 137 biases, making them compact and well-suited for solving a single PDE instance with high accuracy. In contrast, DeepONet requires 112,386 trainable parameters, approximately 25.7 times more than those of PINNs, comprising 62,464 parameters in the branch network, 49,920 in the trunk network, and 2 output biases. The final feature layers of both subnetworks have dimensions 128×2, enabling the simultaneous output of the real and imaginary components. Although this larger parameter budget increases the model complexity, it provides DeepONet with operator-level generalization. Therefore, PINNs are preferable for lightweight, single-instance solving, while DeepONet is advantageous for multi-query and real-time surrogate modeling of parameterized PDEs.

Table B4. Comparison of the number of network parameters of PINNs and DeepONet.

Method	Network architecture	Number of network parameters		
		Weights ($\mathbf{W}$)	Biases ($\mathbf{b}$)	Total trainable parameters $\boldsymbol{\theta}$
PINNs	$[2] + [45] \times 3 + [2]$	$90 + 2025 + 2025 + 90$	$45 + 45 + 45 + 2$	4367
DeepONet	$[100] + [128] + [128] + [128] \times 2$	$12,800 + 16,384 + 32,768$	$128 + 128 + 256$	$62,464 + 49,920 + 2 = 112,386$
	$[2] + [128] + [128] + [128] \times 2$	$256 + 16,384 + 32,768$	$128 + 128 + 256$	

C. Performance of the mixed optimization scheme on other NPDEs

To investigate the generality of the mixed optimization scheme proposed in Section 3.3.1, we select six additional NPDEs as the representative examples to perform numerical experiments. The specific equations and their corresponding exact soliton solutions are detailed as follows:

Case (1): The Boussinesq equation, a renowned nonlinear model for describing shallow-water wave phenomena, is governed by

$$u_{tt} - u_{xx} - u_{xxxx} - 3(u^2)_{xx} = 0, \tag{C1}$$

and admits a one-soliton solution [49]

$$u(x,t) = \frac{1}{2}\mathrm{sech}^2\left(\frac{x-\sqrt{2}t}{2}\right). \tag{C2}$$

Case (2): A fifth-order Korteweg–de Vries (KdV) equation, also known as the Lax equation, is expressed as

$$u_t + 30u^2u_x + 20u_xu_{xx} + 10uu_{xxx} + u_{xxxxx} = 0, \tag{C3}$$

and has the soliton solution [49]

$$u(x,t) = \frac{1}{2}\mathrm{sech}^2\left(\frac{x-t}{2}\right). \tag{C4}$$

Case (3): The (2+1)-dimensional Kadomtsev–Petviashvili (KP) equation, which is widely used in plasma physics, shallow-water wave dynamics, and nonlinear optics, has the form of

$$(u_t + u_{xxx} + 6uu_x)_x + u_{yy} = 0, \tag{C5}$$

whose single-soliton solution is given by [50]

$$u(x,t) = \frac{1}{2}\mathrm{sech}^2\left(\frac{x+y-2t}{2}\right). \tag{C6}$$

Case (4): The complex modified KdV (mKdV) equation, which describes the propagation of short pulses in optical fibers, reads

$$\psi_t + |\psi|^2\psi_x + \psi_{xxx} = 0, \tag{C7}$$

and has the soliton solution [51]

$$\psi(x,t) = \sqrt{6}e^{2i}\mathrm{sech}(x-t). \tag{C8}$$

Case (5): The Chen–Lee–Liu (CLL) equation, a model for simulating the propagation of self-steepening optical pulses in the absence of self-phase modulation, takes the form of

$$i\psi_t + \psi_{xx} + i\psi\psi^*\psi_x = 0, \tag{C9}$$

and admits the soliton solution [52]

$$\psi(x,t) = (1+i)\frac{1+e^{x-2t}}{1+ie^{x-2t}}e^{i(\frac{x}{2}-\frac{5t}{4})}. \tag{C10}$$

Case (6): The Gerdjikov–Ivanov (GI) equation, generally employed to describe Alfvén waves propagating parallel to the ambient magnetic field in plasma physics, is written as

$$i\psi_t + \psi_{xx} - i\psi^2\psi_x^* + \frac{1}{2}\psi^3(\psi^*)^2 = 0, \tag{C11}$$

and has the soliton solution [53]

$$\psi(x,t) = \frac{8e^{i(0.24x+0.0448t)}}{5\cosh(0.32x - 0.1536t) - 10i\sinh(0.32x - 0.1536t)}. \tag{C12}$$

Except for the optimization schemes under consideration, all other experimental conditions are kept identical. The main neural network architecture consists of three hidden layers, with 45 neurons in each hidden layer (the input and output layers may differ slightly depending on the specific equation). The tanh function is adopted as the activation function. For the PDE residual, initial condition, and boundary conditions, we set the training points to $N_{\mathcal{R}} = 10,000$, $N_{\mathcal{I}} = 200$, and $N_{\mathcal{B}} = 400$, respectively. Their weights are set to $w_{\mathcal{R}} = 1$, $w_{\mathcal{I}} = 0.1$, and $w_{\mathcal{B}} = 0.01$. The specific computational domains adopted for each case are listed in Table C1. Numerical results summarized in Table C1 show that the errors for all six tested equations are significantly reduced compared with those obtained using the fixed Adam optimizer. Except for the CLL equation, the runtime is also reduced for the other five equations.

Table C1. Comparison of mixed optimization Scheme 1 (6000 steps Adam + L-BFGS) and Scheme 2 (M-Adam + L-BFGS) for solving the Boussinesq equation (C1), the fifth-order KdV equation (C3), the KP equation (C5), the complex mKdV equation (C7), the CLL equation (C9), and the GI equation (C11).

Equation	Domain	Optimization	$\mathbb{L}^2$-norm errors			Iterations	Time (s)
			u	v	$\lvert\psi\rvert$		
Eq (C1)	$x \in [-10,10], t \in [-3,3]$	Scheme 1	3.062×10^{-2}	–	–	6000 + 1745	79.11
		Scheme 2	1.479×10^{-2}	–	–	3000 + 2146	70.56
Eq (C3)	$x \in [-10,10], t \in [-3,3]$	Scheme 1	2.085×10^{-4}	–	–	6000 + 1446	153.87
		Scheme 2	1.741×10^{-4}	–	–	3000 + 1423	117.25
Eq (C5)	$x \in [-10,10], y \in [-3,3], t \in [-2,2]$	Scheme 1	3.152×10^{-4}	–	–	6000 + 3391	103.78
		Scheme 2	2.774×10^{-4}	–	–	3000 + 3290	87.66
Eq (C7)	$x \in [-10,10], t \in [-3,3]$	Scheme 1	3.302×10^{-4}	2.434×10^{-4}	2.259×10^{-4}	6000 + 2561	72.74
		Scheme 2	4.052×10^{-4}	1.939×10^{-4}	1.906×10^{-4}	3000 + 2818	68.89
Eq (C9)	$x \in [-10,10], t \in [-2,2]$	Scheme 1	6.066×10^{-4}	6.616×10^{-4}	3.510×10^{-4}	6000 + 8304	91.33
		Scheme 2	3.667×10^{-4}	3.760×10^{-4}	2.268×10^{-4}	6600 + 7642	100.97
Eq (C11)	$x \in [-10,10], t \in [-3,3]$	Scheme 1	3.677×10^{-3}	4.160×10^{-3}	1.283×10^{-3}	6000 + 3488	49.26
		Scheme 2	1.418×10^{-3}	1.572×10^{-3}	5.879×10^{-4}	3200 + 3128	46.67

D. Effects of the initial condition with low- and high-frequency noise on the model

In this section, we solve the forward problem to quantify the effects of low-frequency baseline drifts and high-frequency oscillations in the initial conditions on solution reconstruction accuracy, and solve the inverse problem to quantify their effects on parameter identification accuracy. Let $\mathcal{I}_0(x,t)$ denote the clean initial condition. For a prescribed relative noise level ϵ, we perturb the initial condition as

$$\mathcal{I}_0^{\epsilon}(x,t) = \mathcal{I}_0(x,t) + \eta(x,t), \tag{D1}$$

where the perturbation η is normalized to satisfy

$$\|\eta\|_2 = \epsilon \|\mathcal{I}_0\|_2. \tag{D2}$$

Here $\|\cdot\|_2$ denotes the discrete $\mathbb{L}^2$-norm evaluated on the training points.

Low-frequency noise—Low-frequency noise is constructed as a truncated Fourier series with low wavenumbers:

$$\begin{aligned}\eta_{\text{low}}(x,t) = \sum_{m=1}^{M} \Big[& a_m \sin(k_m x + \phi_m)\cos(k_m t) + b_m \cos(k_m x - \phi_m)\sin(k_m t) \\ & + c_m \sin(k_m x + k_m t + \phi_m) + d_m \cos(k_m x - k_m t + \phi_m) \Big],\end{aligned} \tag{D3}$$

where $M = 3$, the wavenumbers are $k_m = \frac{m\pi}{20}$, and the random coefficients satisfy

$$a_m, b_m, c_m, d_m \sim \mathcal{N}(0,1), \quad \phi_m \sim \mathcal{U}(0, 2\pi), \tag{D4}$$

all drawn independently. This generates a smooth perturbation dominated by low spatial and temporal frequencies, mimicking a baseline drift.

High-frequency noise—White Gaussian noise is generated as independent and identically distributed Gaussian random variables at each sampling point:

$$\eta_{\text{white}}(x_i, t_i) \sim \mathcal{N}(0,1), \quad i = 1, \ldots, N, \tag{D5}$$

which corresponds to a nearly flat spectral density in both space and time. High-frequency noise is obtained by removing the low-frequency component from white Gaussian noise:

$$\eta_{\text{high}} = \eta_{\text{white}} - \eta_{\text{low}}. \tag{D6}$$

This construction approximates high-pass filtered noise whose energy is concentrated at high wavenumbers.

For each noise type, the raw perturbation η is rescaled to the prescribed relative level

$$\eta \leftarrow \eta \, \frac{\epsilon \|\mathcal{I}_0\|_2}{\|\eta\|_2}. \tag{D7}$$

This ensures that all three noise types share exactly the same relative $\mathbb{L}^2$ amplitude, so that any difference in the identified parameters λ_1 and λ_2 can be attributed solely to the spectral content of the noise rather than its intensity.

Specifically, we select the NLSE (3.1) as a representative example, with initial conditions given by Eq (3.3). The effects of white Gaussian noise, low-frequency noise, and high-frequency noise at intensities of 1%, 3%, 5%, and 10% on the initial conditions $u(x, -5)$ and $v(x, -5)$ are illustrated in Figure D1.

D.1. Forward problem

We first investigate the forward problem using PINNs. As reported in Section 3.1.1, the relative $\mathbb{L}^2$-norm error for the noise-free baseline case is $O(10^{-3})$. For each noise type and noise level, ten independent numerical experiments are conducted with random seeds ranging from 1001 to 1010.

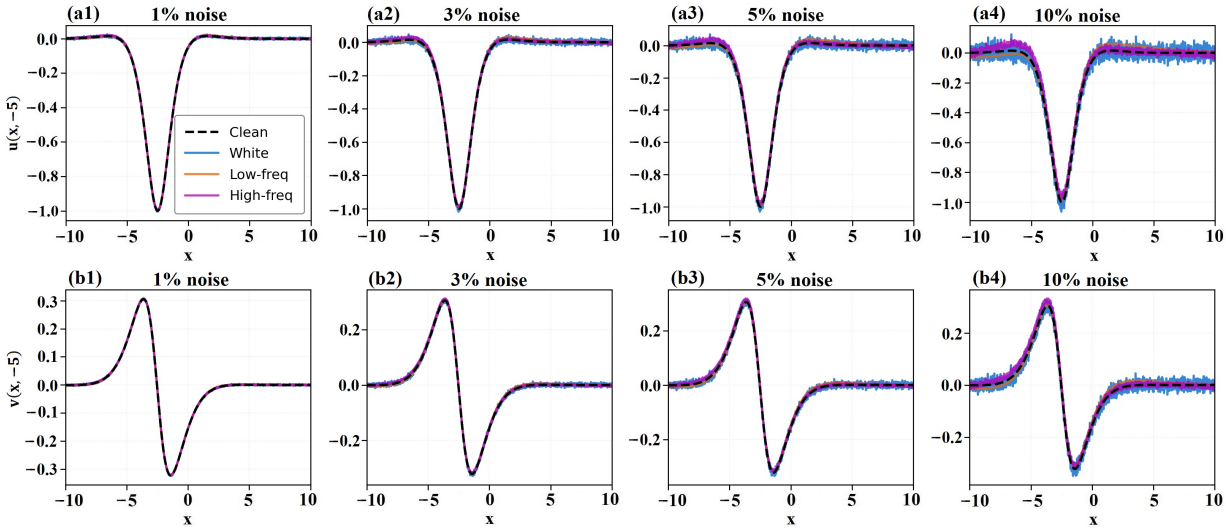


Figure D1. Comparison of white, low-frequency, and high-frequency noise perturbations for the initial conditions of Eq (3.1). The top and bottom rows show $u(x, -5)$ and $v(x, -5)$, respectively. Columns correspond to relative noise levels of 1%, 3%, 5%, and 10% from left to right.

The mean and standard deviation of the relative $\mathbb{L}^2$-norm error for the learned modulus $\left|\hat{\psi}(x,t)\right|$ are summarized in Table D1. These results demonstrate that introducing 1% noise into the initial condition approximately raises the error to $O(10^{-2})$. As the noise level increases, the error grows monotonically for both low-frequency drift and high-frequency oscillation. At the 10% noise level, the mean errors reach $O(10^{-1})$. The mean errors for the two noise types are comparable across all noise levels, which indicates that the forward prediction accuracy for the NLSE is primarily governed by the noise amplitude rather than its spectral content. However, high-frequency oscillation generally yields larger standard deviations.

Table D1. Performance of PINNs in solving Eq (3.1) for the modulus $|\psi(x,t)|$ under varying noise intensities and spectral contents in the initial condition.

	1% noise	3% noise	5% noise	10% noise
Low-frequency drift	$(1.17 \pm 0.38) \times 10^{-2}$	$(3.40 \pm 0.91) \times 10^{-2}$	$(5.74 \pm 1.45) \times 10^{-2}$	$(1.20 \pm 0.23) \times 10^{-1}$
High-frequency oscillation	$(1.17 \pm 0.41) \times 10^{-2}$	$(3.52 \pm 1.36) \times 10^{-2}$	$(5.39 \pm 1.78) \times 10^{-2}$	$(1.06 \pm 0.38) \times 10^{-1}$

We then examine the forward problem using DeepONet. As stated in Section 3.1.2, the relative $\mathbb{L}^2$-norm error of the noise-free baseline is 2.443×10^{-2} evaluated over 2000 test samples. Given the computational overhead, we evaluate the overall error and analyze the resulting trends solely on these 2000 test samples, without repeating the experiments across ten random seeds as performed for PINNs. Table D2 summarizes the performance of DeepONet under noisy training and noisy test data. Three key observations can be drawn: First, the error increases monotonically with the noise level in all cases. Second, noise in the test data degrades prediction accuracy far more severely than noise added to the training dataset. Third, for noisy training data, high-frequency oscillation consistently produces larger

errors than low-frequency drift, which suggests that high-frequency components are more challenging for DeepONet to learn. In contrast, under noisy test data at levels of 3%–10%, low-frequency drift leads to slightly larger errors than high-frequency oscillation.

Table D2. Performance of DeepONet in solving Eq (3.1) for the modulus $|\psi(x,t)|$ under varying noise intensities and spectral contents in the training and test data.

	Noisy training data		Noisy test data	
	Low frequency	High frequency	Low frequency	High frequency
1% noise	3.092×10^{-2}	3.567×10^{-2}	5.294×10^{-2}	5.396×10^{-2}
3% noise	4.194×10^{-2}	7.503×10^{-2}	1.379×10^{-1}	1.216×10^{-1}
5% noise	5.818×10^{-2}	1.043×10^{-1}	2.387×10^{-1}	2.019×10^{-1}
10% noise	6.784×10^{-2}	1.602×10^{-1}	4.351×10^{-1}	3.806×10^{-1}

Based on the above results for noisy forward problems obtained with PINNs and DeepONet, PINNs achieve high accuracy in the noise-free setting. However, their accuracy deteriorates substantially once noise is introduced into the initial condition, and high-frequency noise tends to impair stability more strongly than low-frequency drift. DeepONet exhibits greater sensitivity to initial noise compared with PINNs. In particular, test-time noise, especially low-frequency drift, can significantly degrade its predictive performance.

D.2. Inverse problem

From Table D3, the identification errors of both λ_1 and λ_2 increase monotonically with the noise level. For λ_1, low-frequency and high-frequency perturbations give comparable errors, with the high-frequency case slightly larger at high noise levels. For λ_2, low-frequency baseline drift consistently produces larger errors than high-frequency oscillation. This is because high-frequency components are amplified by the second-order spatial derivative in the PDE residual, mainly affecting λ_1, while low-frequency baseline drift changes the global envelope and has a stronger effect on λ_2.

Table D3. Results of identifying the coefficients $\boldsymbol{\lambda} = \{\lambda_1, \lambda_2\}$ of Eq (4.2) under varying intensities and spectral contents of initial noise.

	Low-frequency drift		High-frequency oscillation	
	Error of $\hat{\lambda}_1$	Error of $\hat{\lambda}_2$	Error of $\hat{\lambda}_1$	Error of $\hat{\lambda}_2$
1% noise	$(1.89 \pm 1.35) \times 10^{-3}$	$(2.09 \pm 1.15) \times 10^{-3}$	$(2.06 \pm 1.40) \times 10^{-3}$	$(1.94 \pm 0.58) \times 10^{-3}$
3% noise	$(5.82 \pm 3.75) \times 10^{-3}$	$(6.40 \pm 3.04) \times 10^{-3}$	$(5.92 \pm 4.49) \times 10^{-3}$	$(5.23 \pm 2.04) \times 10^{-3}$
5% noise	$(9.78 \pm 6.78) \times 10^{-3}$	$(1.48 \pm 1.62) \times 10^{-2}$	$(9.87 \pm 7.84) \times 10^{-3}$	$(9.10 \pm 3.80) \times 10^{-3}$
10% noise	$(1.91 \pm 1.31) \times 10^{-2}$	$(2.12 \pm 1.01) \times 10^{-2}$	$(2.03 \pm 1.60) \times 10^{-2}$	$(1.82 \pm 0.82) \times 10^{-2}$

AIMS Press

© 2027 the Author(s), licensee AIMS Press. This is an open access article distributed under the terms of the Creative Commons Attribution License (https://creativecommons.org/licenses/by/4.0)