Ann. Appl. Math.
doi: 10.4208/aam.OA-2024-0023

Vol. **41**, No. 1, pp. 42-76
February 2025

Deep Neural Network Approaches for Computing the Defocusing Action Ground State of Nonlinear Schrödinger Equation

Zhipeng Chang[1], Zhenye Wen[2,*] and Xiaofei Zhao[2]

[1] *School of Mathematics and Statistics, Wuhan University, Wuhan, Hubei 430072, China*
[2] *Computational Sciences Hubei Key Laboratory, Wuhan University, Wuhan, Hubei 430072, China*

Received 31 October 2024; Accepted (in revised version) 25 January 2025

Dedicated to the celebration of the 70th birthday of Professor Avy Soffer

Abstract. The defocusing action ground state of the nonlinear Schrödinger equation can be characterized via three different but equivalent minimization formulations. In this work, we propose some deep neural network (DNN) approaches to compute the action ground state through the three formulations. We first consider the unconstrained formulation, where we propose the DNN with a shift layer and demonstrate its necessity towards finding the correct ground state. The other two formulations involve the L^{p+1}-normalization or the Nehari manifold constraint. We enforce them as hard constraints into the networks by further proposing a normalization layer or a projection layer to the DNN. Our DNNs can then be trained in an unconstrained and unsupervised manner. Systematical numerical experiments are conducted to demonstrate the effectiveness and superiority of the approaches.

AMS subject classifications: 35Q55, 68T07, 81-08, 81Q05

Key words: Nonlinear Schrödinger equation, action ground state, deep neural network, shift layer, normalization layer, projection layer.

*Corresponding author.
Emails: changzhipeng@whu.edu.cn (Z. Chang), wenzhy@whu.edu.cn (Z. Wen), matzhxf@whu.edu.cn (X. Zhao)

1 Introduction

The nonlinear Schrödinger equation (NLS) is widely applied in fields such as quantum physics, nonlinear optics, fluid dynamics, and plasma physics [3, 7, 11, 15, 16, 19, 40, 43, 44, 49, 51, 54]. Under the influence of an external potential field, the NLS equation reads as:

$$i\partial_t\psi(\mathbf{x},t)=-\frac{1}{2}\Delta\psi(\mathbf{x},t)+V(\mathbf{x})\psi(\mathbf{x},t)+\beta|\psi(\mathbf{x},t)|^{p-1}\psi(\mathbf{x},t),\quad t>0, \tag{1.1a}$$

$$\mathbf{x}=(x_1,\cdots,x_d)^\top\in\mathbb{R}^d, \tag{1.1b}$$

where $\beta\in\mathbb{R}$ and $p>1$ are given parameters, $\psi(\mathbf{x},t):\mathbb{R}^d\times\mathbb{R}\to\mathbb{C}$ is the unknown complex-valued wave function. Here, $V(\mathbf{x})$ is a real-valued potential function. A commonly employed example of such a potential is the harmonic oscillator potential, defined as

$$V(\mathbf{x})=\frac{1}{2}\sum_{j=1}^{d}\gamma_j^2x_j^2\quad\text{ with }\quad\gamma_j\geq 0.$$

The parameter β characterizes the strength of the nonlinear self-interaction, with $\beta>0$ corresponding to a defocusing interaction and $\beta<0$ corresponding to a focusing interaction. The standing wave/stationary solution of the NLS equation (1.1) is considered key quantities for understanding the evolution of wave systems. By setting $\psi(\mathbf{x},t)=\mathrm{e}^{i\omega t}\phi(\mathbf{x})$ in (1.1), the stationary solution $\phi(x)$ satisfies the following elliptic equation:

$$-\frac{1}{2}\Delta\phi(\mathbf{x})+V(\mathbf{x})\phi(\mathbf{x})+\beta|\phi(\mathbf{x})|^{p-1}\phi(\mathbf{x})+\omega\phi(\mathbf{x})=0,\quad\mathbf{x}\in\mathbb{R}^d, \tag{1.2}$$

where $\omega\in\mathbb{R}$ represents the given chemical potential. In fact, there can be infinitely many nontrivial solutions ($\phi(\mathbf{x})\not\equiv 0$) that satisfy (1.2) [12, 50]. Among these nontrivial solutions, the one that minimizes the *action functional*

$$S_\omega(\phi):=\frac{1}{2}\|\nabla\phi\|_{L^2}^2+\int_{\mathbb{R}^d}V|\phi|^2d\mathbf{x}+\frac{2\beta}{p+1}\|\phi\|_{L^{p+1}}^{p+1}+\omega\|\phi\|_{L^2}^2 \tag{1.3}$$

is referred to as the *action ground state*, denoted by ϕ_g [11, 12].

The action ground state is of great importance in mathematical and physical studies [4, 11, 15, 19, 24, 25, 37, 38, 53]. Particularly, it is needed in the computations of the multichannel dynamics in NLS [47–49]. Also, the recent works [22, 30, 37] reveal its non-equivalence with the energy ground state, making its computation to own more independent interests. It can be rigorously defined as follows. Note that

the variation of $S_\omega(\phi)$ is given by

$$\frac{\delta S_\omega(\phi)}{\delta\overline{\phi}}=\left(-\frac{1}{2}\Delta+V+\beta|\phi|^{p-1}+\omega\right)\phi=:H_\phi(\phi),$$

and so the elliptic equation (1.2) can be expressed as $H_\phi(\phi)=0$. With the function space

$$X=\left\{\phi\in H^1(\mathbb{R}^d):\int_{\mathbb{R}^d}V(\mathbf{x})|\phi(\mathbf{x})|^2d\mathbf{x}<\infty\right\},$$

the action ground state can be given as originally in [11] by the minimization with a natural constraint:

$$\phi_g\in\arg\min\{S_\omega(\phi)\ :\ \phi\in X\backslash\{0\},\ H_\phi(\phi)=0\}. \tag{1.4}$$

Theoretically, (1.4) can be equivalently defined by using a Nehari manifold constraint [4,13,24,39], which we refer as the Nehari constrained formulation. Alternatively, the recent work [39] characterizes the problem (1.4) equivalently by minimizing a quadratic functional on an L^{p+1} unit sphere, leading to the L^{p+1}-normalization formulation. Moreover, in the defocusing case, [39] has shown that the constraint in (1.4) can be disregarded, allowing for an unconstrained minimization formulation. Thus, the action ground state for the defocusing NLS has three different but equivalent formulations. Based on these formulations, a series of numerical methods have been proposed, including the classical gradient flow method, also known as the imaginary time evolution method, and several constrained optimization techniques [38, 39, 53]. These methods are powerful but classical discretizations, which inevitably suffers from problems such as the curse-of-dimensionality and the multiple physical parameter-dependence.

Deep neural networks (DNNs) have become a powerful tools in many areas, and have already been extensively applied to solve scientific computing problems. Despite the fact that there are currently no effective DNN methods to solve the action ground state, some feasible DNN methods have emerged to tackle its "twin counterpart", the energy ground state. For detailed reviews of the energy ground state and its relationship with the action ground state, we refer the readers to [1, 2, 7, 9, 13, 14, 18, 20, 21, 27, 35, 55–57]. The existing DNNs for the energy ground states include some supervised learning methods [5, 6, 34], while in practical problems, obtaining the labeled datasets can be very costly. Therefore, unsupervised learning methods based on DNNs would be preferred. Recently, a normalized deep neural network (norm-DNN) is proposed for computing the energy ground states of Bose-Einstein condensation via the minimization of the Gross-Pitaevskii energy functional under unitary mass normalization [8]. The idea of norm-DNN can also be extended

to solve other constrained minimization problems, for instance the initial-terminal value problems [17] and of course the action ground state.

In this paper, we consider the defocusing action ground states under three different formulations and design corresponding DNNs based on the relevant properties of the action ground states. Firstly, for the unconstrained formulation of the action ground state, we propose a DNN that incorporates key techniques such as a shift layer and Gaussian pre-training. Numerical experiments demonstrate that these techniques effectively mitigate the "local non-positivity" issue in the approximate solution and accelerate convergence to the ground state solution. Then, for the other two constrained formulations, we advocate the introduction of a normalization layer and a projection layer to handle the constraints, thus proposing the L^{p+1}-normalized DNN and the Nehari projected DNN. Systematical investigations are carried out through extensive numerical experiments, and the results show that compared to the conventional DNNs with soft-type penalties in the loss function, our novel DNN approaches have better accuracy and efficiency in solving the constrained action ground state problems. Comparisons are also made between the three proposed DNN approaches, and the L^{p+1}-normalized DNN is identified as the most effective one. At last, some more techniques are introduced to further improve the performance the proposed DNN methods.

The rest of this paper is structured as follows. Section 2 provides the necessary preliminaries for the problem. Section 3 details the development of the DNN with a shift layer. The design of the normalization layer to address the L^{p+1}-normalization constraint is discussed in Section 4. Section 5 introduces the DNN designed for the Nehari constrained formulation. Section 6 compares the proposed DNN approaches and explores some applications and extensions. Some conclusions are summarized in Section 7.

2 Preliminary

In this section, we shall first give a brief overview of three equivalent formulations of the action ground state, and then go through some features of the action ground state, as well as the traditional numerical methods. Finally, we present the backgrounds for the classical deep neural networks (DNNs).

2.1 Formulations for action ground state

Compared to the original definition of the action ground state (1.4), which refers to the solutions of the elliptic equation (1.2) that minimize the action functional $S_\omega(\phi)$ (1.3), in the defocusing case, the three equivalent formulations of the action

ground state are typically used in the design of numerical methods [38,39,53]. Prior to formally stating the details of these formulations, we first present a fundamental assumption regarding the ground state problem:

Assumption 2.1. Assume that the potential function $V(\mathbf{x})\geq 0$ $(\forall \mathbf{x}\in\mathbb{R}^d)$ and $\lim_{|\mathbf{x}|\to\infty}V(\mathbf{x})=\infty$. Let $\beta>0$, $1<p<(d+2)/(d-2)$ for $d\geq 3$ and $1<p<\infty$ for $d=1,2$, and $\omega<-\lambda_0$, where

$$\lambda_0:=\inf\left\{\frac{1}{2}\|\nabla u\|_{L^2}^2+\int_{\mathbb{R}^d}V|u|^2d\mathbf{x}\colon \|u\|_{L^2}=1\right\}.$$

Under the assumptions in Assumption 2.1 about $V(x)$ and other parameters, the action functional (1.3) is uniformly bounded in X, and the constraint in (1.4) can be removed, meaning that the action ground state can be obtained by solving an unconstrained minimization problem [39]. Specifically, the unconstrained formulation of the action ground state is described by the following unconstrained minimization problem:

$$\phi_g\in\operatorname{argmin}\{S_\omega(\phi)\ :\ \phi\in X\backslash\{0\}\}. \tag{2.1}$$

The second formulation, referred to as the L^{p+1}-normalization formulation, asserts that the action ground state can also be obtained by minimizing a quadratic functional on an L^{p+1} unit sphere [38, 39]. Let's denote the L^{p+1} unit sphere by $\mathcal{S}_{p+1}=\{u\in X:\|u\|_{L^{p+1}}=1\}$ and present a quadratic functional (i.e., the quadratic part in the action functional (1.3)) as

$$Q(u):=\frac{1}{2}\|\nabla u\|_{L^2}^2+\int_{\mathbb{R}^d}V|u|^2d\mathbf{x}+\omega\|u\|_{L^2}^2. \tag{2.2}$$

By solving the quadratic minimization problem

$$u_*\in\operatorname{argmin}\{Q(u)\ :\ u\in\mathcal{S}_{p+1}\}, \tag{2.3}$$

then the action ground state ϕ_g can be obtained through

$$\phi_g(\mathbf{x})=\left(\frac{Q(u_*)}{-\beta}\right)^{\frac{1}{p-1}}u_*(\mathbf{x}). \tag{2.4}$$

The third formulation characterizes the action ground state by minimizing the action functional (1.3) under a Nehari manifold constraint, which is referred to as the Nehari constrained formulation [4, 13, 24, 39]. The Nehari functional is defined as the L^2-inner product of Eq. (1.2) with ϕ:

$$K_\omega(\phi):=\frac{1}{2}\|\nabla\phi\|_{L^2}^2+\int_{\mathbb{R}^d}V|\phi|^2d\mathbf{x}+\beta\|\phi\|_{L^{p+1}}^{p+1}+\omega\|\phi\|_{L^2}^2,$$

and the set of nontrivial ϕ in $X\backslash\{0\}$ that satisfy $K_\omega(\phi)=0$ forms the so-called Nehari manifold:

$$\mathcal{N}_\omega := \{\phi \in X\backslash\{0\}, K_\omega(\phi)=0\},$$

which includes all nontrivial solutions of (1.2). The action ground state defined in (1.4) can be expressed as the minimizer of action functional (1.3) on Nehari manifold $\mathcal{N}_\omega$, i.e.,

$$\phi_g \in \operatorname{argmin}\{S_\omega(\phi) \,:\, \phi \in \mathcal{N}_\omega\}. \tag{2.5}$$

For the Nehari manifold, we have the following proposition:

Proposition 2.1 ([38]). *When $\omega < -\lambda_0$, for any $\phi \in H^1(\mathbb{R}^d)\backslash\{0\}$, there exists a unique $\sigma_\omega(\phi)>0$ such that $K_\omega(\sigma_\omega(\phi)\phi)=0$, where*

$$\sigma_\omega(\phi) = \left[\frac{\|\nabla\phi\|_{L^2}^2 + 2\int_{\mathbb{R}^d} V|\phi|^2 d\mathbf{x} + 2\omega\|\phi\|_{L^2}^2}{-2\beta\|\phi\|_{L^{p+1}}^{p+1}}\right]^{\frac{1}{p-1}}. \tag{2.6}$$

This proposition indicates that any $\phi \in X\backslash\{0\}$ can be projected onto the Nehari manifold $\mathcal{N}_\omega$ using the projection operator defined in (2.6), which allows us to effectively manage the Nehari manifold constraint in problem (2.5).

2.2 Features of action ground state and traditional numerical method

Let us provide some important mathematical features for the action ground states from the theoretical studies [4, 22, 24, 39]. Based on the assumptions regarding the potential function $V(\mathbf{x})$ and other parameters in Assumption 2.1, the action ground state satisfies:

(i) If the potential function $V(\mathbf{x})$ is smooth, the ground state $\phi_g \in C^\infty$ decays exponentially fast to zero when $|\mathbf{x}| \to \infty$;

(ii) When the potential function $V(\mathbf{x})$ is isotropic, with a shift in the phase $\mathrm{e}^{i\xi}\phi_g$ for any $\xi \in \mathbb{R}$, action ground state ϕ_g can be chosen non-negative.

We will design the DNN approaches based on these features to achieve accurate and efficient approximations of the action ground state, with detailed specifications provided in the subsequent sections. Before that, we review a traditional method for solving the action ground state, known as the gradient flow method. It will serve as the benchmark for reference action ground state solutions for the subsequent DNN approaches. Based on the unconstrained formulation (2.1), a gradient

flow method with backward-forward Euler temporal discretization scheme (GF-BF) proposed in [39] reads as

$$\frac{\phi^{n+1}-\phi^n}{\tau}=\frac{1}{2}\Delta\phi^{n+1}-\alpha_n\phi^{n+1}+\left(\alpha_n-V-\omega-\beta|\phi^n|^{p-1}\right)\phi^n,\quad n\geq 0, \tag{2.7}$$

where $\tau>0$ denotes the time step and $\alpha_n\geq 0$ represents a stabilization factor. The stabilization factor $\alpha_n\geq 0$ aids in the descent of the action functional, allowing for a larger time step τ to be employed. The computational domain is truncated to a bounded region U and we adopt

$$\alpha_n=\frac{1}{2}\max\left\{0,\max_{\mathbf{x}\in U}\left(V(\mathbf{x})+\omega+\beta|\phi^n(\mathbf{x})|^{p-1}\right)\right\},$$

following the recommendation in [39].

2.3 Standard DNN and its training

DNNs are a class of artificial neural networks composed of multiple layers, designed to mimic the structure and function of the human brain for processing and analyzing complex data. Due to their powerful capability to handle high-dimensional and large-scale data, DNN have achieved remarkable success in fields like computer vision, natural language processing, and speech recognition [10,32,33,42,45]. Mathematically, a DNN with L hidden layers can be represented as a composite function

$$f(\mathbf{y}_0;\theta)=\mathbf{F}_{L+1}\circ\sigma\circ\mathbf{F}_L\circ\sigma\circ\cdots\circ\mathbf{F}_2\circ\sigma\circ\mathbf{F}_1(\mathbf{y}_0),\quad \theta:=\{W_l,b_l\}, \tag{2.8}$$

where the affine transformation of the l-th layer $\mathbf{F}_l$ is defined as:

$$\mathbf{F}_l(\mathbf{y}_{l-1})=W_l\mathbf{y}_{l-1}+b_l,\quad \mathbf{y}_{l-1}\in\mathbb{R}^{n_{l-1}},\quad 1\leq l\leq L+1,$$

with $W_l\in\mathbb{R}^{n_l\times n_{l-1}}$ representing the weight matrix and $b_l\in\mathbb{R}^{n_l}$ being the bias vector for $n_l\in\mathbb{N}_+$. The final layer $\mathbf{F}_{L+1}$ serves as the output layer and $\mathbf{F}_l$ for $1\leq l\leq L$ are called the hidden layers, with n_l neurons in each layer. Specifically, n_0 and n_{L+1} represent the dimensions of input and output, respectively. Furthermore, $\sigma:\mathbb{R}\to\mathbb{R}$ serves as the activation function that acts on each component of $\mathbf{F}_l$, i.e., $\sigma\circ\mathbf{F}\in\mathbb{R}^{n_l}$. Selecting an appropriate activation function is a crucial topic for specific problems, and we will elaborate on this in detail in Section 6. For simplicity, we will consider the fully connected neural network (FCNN), where each hidden layer contains the same number of neurons, i.e., $n_1=n_2=\cdots=n_L=W$. In this setting, the 'depth' of the network is defined as the total number of hidden layers L, and the 'width' refers to the number of neurons W in each hidden layer. This uniform configuration allows for a more straightforward analysis of network properties.

To apply the deep learning algorithm, we need to define the following optimization problem:

$$\min_{\theta}\{\text{Loss}(\theta)\}, \tag{2.9}$$

where $\text{Loss}(\theta)$ represents the optimization objective, also known as the loss function. Since the three different formulations of the action ground state are all minimization problems, the functional that needs to be minimized in each formulation can naturally serve as the loss function. In other words, the action functional S_ω (1.3) or quadratic functional Q (2.2) is employed as the loss function $\text{Loss}(\theta)$ for training the DNNs and optimizing the parameters θ. Note that we must rely on numerical integration to compute the S_ω (1.3) and Q (2.2) in practical applications, which necessitates truncating the entire space $\mathbb{R}^d$ to a bounded region U and discretizing the integral accordingly. Due to the exponential decay of the action ground state in the far field, the truncation error would be negligible. Here, for the three DNN approaches, we select a fixed, uniformly distributed grid of points, $\{\mathbf{x}_j\}_{j=1}^N \subset U$, over the domain U as the training set. Then, the $\text{Loss}(\theta)$ is the Riemann sum and the quadrature error can be very small.

In machine learning, the primary approach to solve (2.9) is the gradient descent method: update the parameters as $\theta^{n+1}=\theta^n-\tau\nabla\text{Loss}(\theta^n)$ for $n\geq 0$, where θ^0 is the initial guess of parameters and τ denotes the learning rate. We use the Xavier method (also known as Glorot initialization) for parameter initialization, which reduces issues related to vanishing or exploding gradients, thereby stabilizing the training process [26]. Regarding the choice of learning rate, we adopt a commonly used strategy that enhances training stability and accelerates convergence: gradually decreasing the learning rate during the training process [28, 45]. Unless otherwise stated, the initial learning rate is set to $\tau=0.001$ in this paper, decaying by a factor of 0.99 every 100 steps. To avoid the additional troubles so that we can focus on the development of the network in this paper, the Adam optimizer [31] will be utilized for (2.9), which is a special optimization method widely considered in deep learning. Given a threshold $tol>0$, the stopping criterion for training is defined as

$$\left|\sum_{k=100(j+1)}^{100(j+2)}\text{Loss}(\theta^k)-\sum_{k=100j}^{100(j+1)}\text{Loss}(\theta^k)\right| \Big/ \left|\sum_{k=100j}^{100(j+1)}\text{Loss}(\theta^k)\right| < tol, \quad j=0,1,2,\cdots, \tag{2.10}$$

which indicates that the mean value of the loss function over adjacent 100 iterations stabilizes. To quantify the accuracy of the proposed DNN, we shall compute the relative error as

$$\text{error} := \|\phi_\theta-\phi_g\|_2 / \|\phi_g\|_2, \tag{2.11}$$

which measures the difference between ϕ_θ and ϕ_g. Here, ϕ_θ represents the numerical solution given by DNN approach, and ϕ_g refers to the exact solution ϕ_g obtained by the high-resolution GF-BF method. The L^2-norm in (2.11) is computed by taking the square root of the Riemann sum across a significantly finer grid than that used for training. This finer grid serves as the test set, and the number of test points is 2048 for one-dimensional (1D) problems and 400×400 for two-dimensional (2D) problems.

3 DNN approach based on the unconstrained formulation

In this section, we consider the unconstrained formulation (2.1) and design a DNN with a shift layer to solve for the action ground state.

3.1 DNN with a shift layer

The unconstrained formulation (2.1) enables direct minimization of the action functional (1.3) without considering the constraint. Based on the property that the ground state can be chosen as non-negative through a phase shift, we consider the following **DNN with a shift layer**:

$$\phi_{\mathcal{T}}(\mathbf{x};\theta)=\mathcal{T}\circ\mathbf{F}_{L+1}\circ\sigma\circ\mathbf{F}_L\circ\sigma\circ\cdots\circ\mathbf{F}_2\circ\sigma\circ\mathbf{F}_1(\mathbf{x}),\quad \mathbf{x}\in\mathbb{R}^d. \tag{3.1}$$

The distinct layer $\mathcal{T}$ is referred to as the *shift layer* (also introduced in [8]):

$$\mathcal{T}(\mathbf{y})=\mathbf{y}-\min_{\mathbf{x}}(\mathbf{y}), \tag{3.2}$$

where $\mathbf{y}=\mathbf{y}(\mathbf{x}):\mathbb{R}^d\to\mathbb{R}^{n_{L+1}}$ and the minimization is acting on each component of $\mathbf{y}$. It provides a non-negativity restriction and facilitates the convergence of the DNN to the ground state rather than the excited state during the training process, which will be clarified by subsequent numerical experiments. The activation function $\sigma=\tanh$ is utilized to the DNN with a shift layer (3.1), owing to its smoothness and differentiability, which contribute to its popularity in neural network architectures. Based on the unconstrained formulation (2.1), the optimization objective in this approach is the action functional (1.3), thus we let $\text{Loss}:=S_\omega(\phi_{\mathcal{T}})$. As mentioned in Section 2, we truncate the computational domain to a bounded domain U and

discretize it. In practical, the loss that we will use in (2.9) is defined by

$$\text{Loss}(\theta)=\frac{|U|}{N}\sum_{j=1}^{N}\Big[\frac{1}{2}|\nabla\phi_{\mathcal{T}}(\mathbf{x}_j;\theta)|^2+V(\mathbf{x}_j)|\phi_{\mathcal{T}}(\mathbf{x}_j;\theta)|^2+\frac{2\beta}{p+1}|\phi_{\mathcal{T}}(\mathbf{x}_j;\theta)|^{p+1} +\omega|\phi_{\mathcal{T}}(\mathbf{x}_j;\theta)|^2\Big], \tag{3.3}$$

where $|U|$ denote the measure of U and $\{\mathbf{x}_j\}_{i=1}^N\subset U$ represents the discrete quadrature points for training.

3.2 Numerical experiment

Here we would like to start with the 1D case to elucidate the principles and performance of our proposed DNN with a shift layer (3.1). By truncating the domain to $(-I,I)$, we construct a training set consisting of equidistant nodes defined as $x_j=-I+2Ij/N$, $j=0,1,\cdots,N$, the DNN with a shift layer (3.1) is then employed to solve the following problem:

$$\theta^*=\arg\min_{\theta}\frac{2I}{N}\sum_{j=1}^{N}\Big[\frac{1}{2}|\nabla\phi_{\mathcal{T}}(x_j;\theta)|^2+V(x_j)|\phi_{\mathcal{T}}(x_j;\theta)|^2+\frac{2\beta}{p+1}|\phi_{\mathcal{T}}(x_j;\theta)|^{p+1} +\omega|\phi_{\mathcal{T}}(x_j;\theta)|^2\Big]. \tag{3.4}$$

Through numerical experiments, we will demonstrate the necessity of the shift layer $\mathcal{T}$ (3.2), the acceleration effects of Gaussian pre-training, and the influence of network architecture on approximation performance.

3.2.1 Necessity of shift layer

Since the parameters in DNNs are all real-valued, the action ground state would be determined up to a phase shift and a valid approximation of the action ground state by the DNN should be either $|\phi_g|$ or $-|\phi_g|$. Due to the inherent randomness in the initialization and optimization algorithms, standard DNNs (2.8) may result in that some part of ϕ_θ converges to $|\phi_g|$ while some other part converges to $-|\phi_g|$, which leads to a local minima/excited state instead of the correct action ground state. The introduction of the shift layer $\mathcal{T}$ is aimed at overcoming this issue, and its necessity will be demonstrated through the following numerical experiment. We examine a 1D example to illustrate the performance of DNNs, with/without the shift layer $\mathcal{T}$ (3.2), in approximating the ground state solution.

Example 3.1. Let $d=1$, $p=3$, $\beta=10$, $\omega=-30$ in NLS (1.1), employ the harmonic oscillator potential $V(x)=\frac{1}{2}x^2$, and fix the computational domain $U=(-12,12)$. The exact action for the ground state is $S_\omega(\phi_g)=-371.1406$, which can be obtained using the GF-BF method.

The standard DNN (2.8) without the shift layer in the 1D case can be written as follows:

$$f(x;\theta)=\mathbf{F}_{L+1}\circ\sigma\circ\mathbf{F}_L\circ\sigma\circ\cdots\circ\mathbf{F}_2\circ\sigma\circ\mathbf{F}_1(x),\quad x\in\mathbb{R},\tag{3.5}$$

and our proposed DNN with a shift layer (3.1) is expressed as:

$$\phi_{\mathcal{T}}(x;\theta)=\mathcal{T}\circ\mathbf{F}_{L+1}\circ\sigma\circ\mathbf{F}_L\circ\sigma\circ\cdots\circ\mathbf{F}_2\circ\sigma\circ\mathbf{F}_1(x),\quad x\in\mathbb{R},\tag{3.6}$$

where $\mathcal{T}(y)=y-\min_x(y)$, $y=y(x):\mathbb{R}\to\mathbb{R}$. Setting the activate function as $\sigma=tanh$ and the hyper-parameters as $L\in\{1,2,3,4\}$, $W\in\{10,50,70\}$, $tol=10^{-6}$, $N=128$, we apply both the DNNs with/without the shift layer $\mathcal{T}$ (3.2) to compute the action ground state for Example 3.1.

We present the relative errors (2.11) for the DNNs with/without a shift layer in Table 1 and display the corresponding action functional values $S_\omega(\phi_\theta)$ in Table 2. As shown in Table 1, the relative error (2.11) for the standard DNN (3.5) without the shift layer $\mathcal{T}$ always exceed one, indicating an invalid approximation of the action ground state. In contrast, when the shift layer $\mathcal{T}$ is applied, the relative error of $\phi_{\mathcal{T}}$ can reduce to 5.03×10^{-3}. From Table 2, we find that the action functional value $S_\omega(\phi_\theta)$ of the DNN with a shift layer (3.6) is smaller and closer to the action minimum $S_\omega(\phi_g)$, while $S_\omega(\phi_\theta)$ of the standard DNN (3.5) is lager. These results indicate that the standard DNN (3.5), which does not include the shift layer $\mathcal{T}$, produces some excited states rather than the ground states. Besides, Fig. 1 shows the exact and numerical solutions as well as the associated pointwise errors obtained from the DNNs with/without a shift layer for $L=3$, $W=50$. Fig. 1(a) reveals that the output of the standard DNN (3.5) is "locally non-positive": for $x<0$, the standard DNN converges to $|\phi_g|$, while for $x>0$, it converges to $-|\phi_g|$, resulting in a jump at $x=0$. When the shift layer $\mathcal{T}$ is applied, the output of the DNN with a shift layer $\phi_{\mathcal{T}}$ (3.6) is constrained to be non-negative. Thus, the pointwise error of the DNN with a shift layer is significantly lower than that of the standard DNN (3.5), as shown in Fig. 1(b). The numerical experiments show that the incorporation of the shift layer $\mathcal{T}$ in the DNN (3.1) avoids the phenomenon of "locally non-positive" in the numerical solution and guides the training of the DNN to converge to the ground state rather than the excited state, highlighting its crucial role in accurately capturing the ground state. The above conclusion also holds for the following two DNNs introduced later. Consequently, in the subsequent network configurations, we also add the shift layer $\mathcal{T}$ (3.2).

Table 1: Relative error (2.11) for Example 3.1 with or without shift layer $\mathcal{T}$.

Error(2.11)	L \ W	10	50	70
with $\mathcal{T}$	1	3.13E-2	4.55E-2	3.74E-2
	2	1.51E-2	2.93E-2	1.34E-2
	3	1.02E-2	5.13E-3	1.73E-2
	4	6.34E-3	8.18E-3	5.03E-3
without $\mathcal{T}$	1	2.00E+0	1.72E+0	2.00E+0
	2	1.40E+0	1.40E+0	2.00E+0
	3	1.40E+0	1.40E+0	1.42E+0
	4	1.40E+0	1.42E+0	1.40E+0

Table 2: Numerical action $S_\omega(\phi_\theta)$ for Example 3.1 with or without shift layer $\mathcal{T}$ ($S_\omega(\phi_g)=-371.1406$).

$S_\omega(\phi_\theta)$	L \ W	10	50	70
with $\mathcal{T}$	1	-370.8961	-370.6021	-370.7846
	2	-371.0802	-370.8877	-371.0891
	3	-371.1089	-371.1332	-371.0541
	4	-371.1285	-371.0835	-371.1291
without $\mathcal{T}$	1	-370.0822	-370.0549	-370.0848
	2	-370.1351	-370.0336	-370.0342
	3	-370.1357	-370.1370	-370.1346
	4	-370.1315	-370.0715	-370.1387

3.2.2 Gaussian pre-training

Experience with traditional methods for solving the action ground state suggests that the ground state solution is localized and smooth, resembling a Gaussian function. Consequently, Gaussian functions are often employed as initial data for traditional methods to compute the action ground state [38, 39]. We borrow this idea to the training of the DNN with a shift layer (3.1). Specifically, prior to the formal training phase, a Gaussian function is used for pre-training to obtain a better initial value θ^0 for solving (3.4). In Example 3.1, we set the hyper-parameters as $L=2$, $W=70$, $tol=10^{-6}$, $N=128$. The DNN with a shift layer (3.1), initialized with Xavier initialization, is first trained for 1000 iterations to fit a Gaussian function $\phi_0=2\mathrm{e}^{-x^2/10}$ and the loss function for the Gaussian pre-training is defined as:

$$\mathrm{Loss}(\theta):=\frac{1}{N}\sum_{j=1}^{N}[\phi_{\mathcal{T}}(x_j;\theta)-\phi_0(x_j)]^2.$$

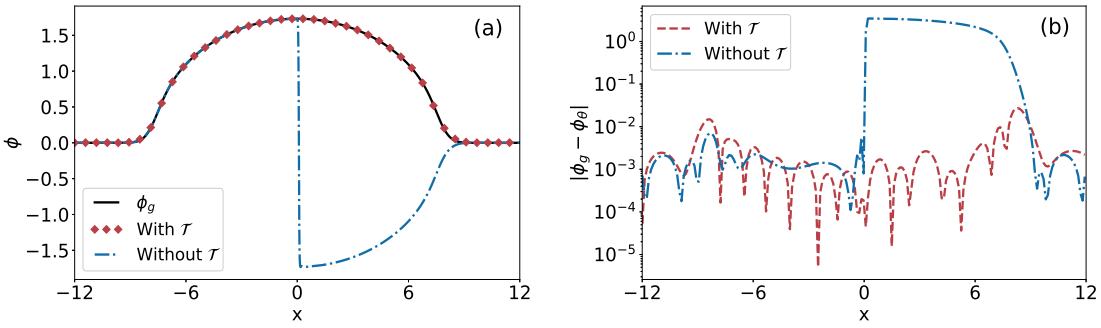

Figure 1: Results for Example 3.1 with L=3, W=50: (a) profiles of the exact solution ϕ_g and numerical solutions with or without $\mathcal{T}$; (b) pointwise errors for the numerical solutions.

Subsequently, the normal training process begins to approximate the ground state.

We evaluate the effects of Gaussian pre-training for DNN with a shift layer (3.1), denoting the two networks as $\phi_{\mathcal{T}}^G$ (with pre-training) and $\phi_{\mathcal{T}}$ (without pre-training). Table 3 presents the total number of iterations during training, the computational time (in seconds) and the relative error (2.11) for $\phi_{\mathcal{T}}^G$ and $\phi_{\mathcal{T}}$. It is evident that Gaussian pre-training enhances both the efficiency and the accuracy significantly. Fig. 2 presents several relevant results. From Fig. 2(a), it can be seen that Gaussian pre-training provides an initial value that is closer to the ground state compared to Xavier initialization, which explains the excellent results achieved through its application. As shown in Figs. 2(e) and (f), after performing the Gaussian pre-training, the DNN with a shift layer can achieve a faster decrease in both the relative error (2.11) and the action functional value. We refer to the DNN that uses the Gaussian pre-training but without the shift layer $\mathcal{T}$ as ϕ^G. It is important to emphasize that while Gaussian pre-training improves performance, it cannot substitute for the role of the shift layer $\mathcal{T}$, as it does not prevent the numerical solution from exhibiting "locally non-positive" behavior, as illustrated in Figs. 2(b) and (d). Therefore, we will apply the shift layer and perform Gaussian pre-training for all subsequent DNNs.

Table 3: Number of iterations, computational time and error (2.11) for DNN (3.1) with or without Gaussian pre-training.

	Iterations	Time	Error (2.11)
$\phi_{\mathcal{T}}$	9300	54s	1.34E-2
$\phi_{\mathcal{T}}^G$	6100	35s	3.67E-3

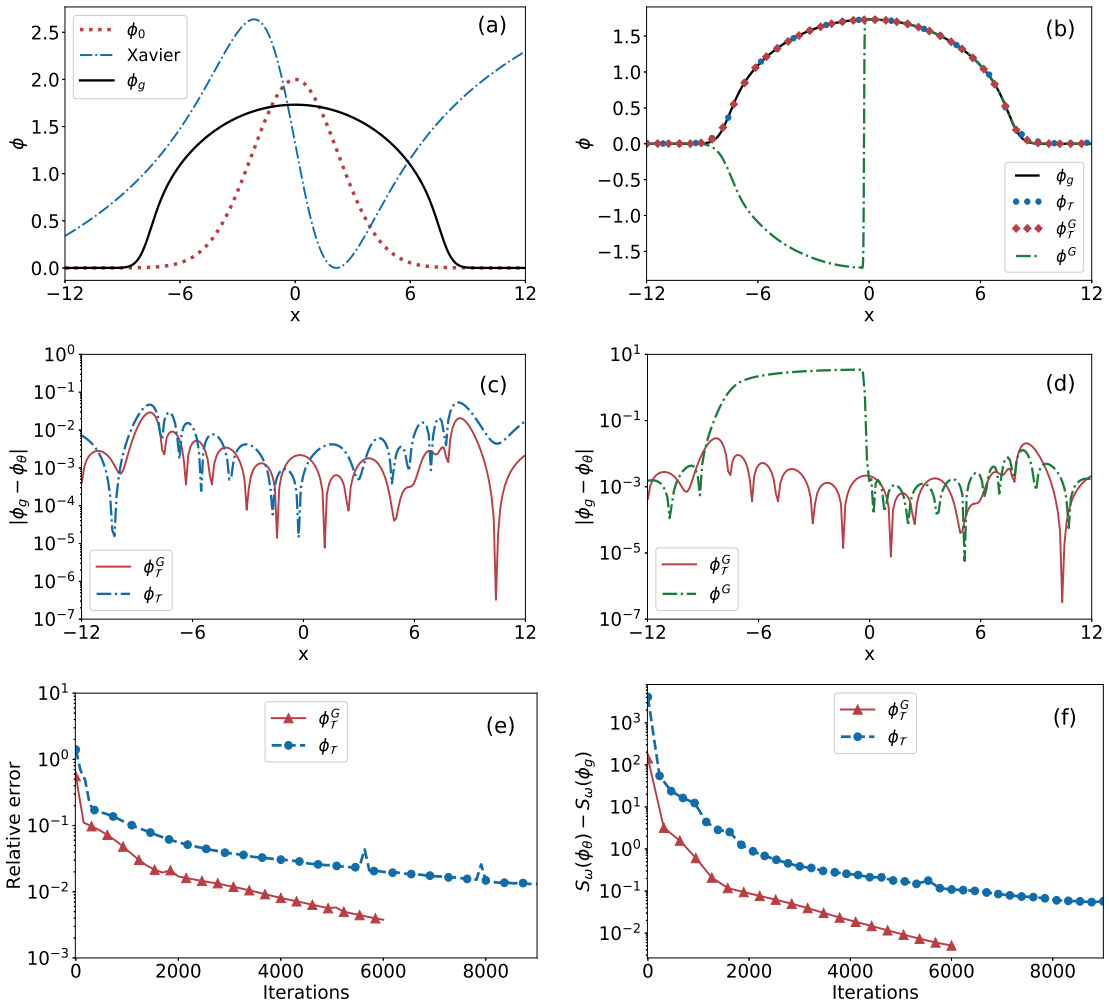

Figure 2: Results for Example 3.1 with $L=2$, $W=70$: (a) initial profile of $\phi_\mathcal{T}$ from Gaussihan pre-training or Xavier method; (b) exact solution ϕ_g, numerical solutions $\phi_\mathcal{T}^G$ with $\mathcal{T}$ and pre-training, $\phi_\mathcal{T}$ with $\mathcal{T}$ but without pre-training, ϕ^G with pre-training but without $\mathcal{T}$; (c), (d) pointwise errors; (e), (f) the change of error (2.11) and $S_\omega(\phi_\theta)-S_\omega(\phi_g)$ during the iterations (in logarithmic scale).

3.2.3 Influence of network architecture

Then we investigate the impact of the network architecture on the DNN with a shift layer (3.1), with the hyper-parameters set as $L\in\{1,2,3,4\}$, $W\in\{10,50,70\}$, $tol=10^{-7}$, $N=256$. Table 4 presents the errors (2.11) obtained by DNNs with different depths and widths. As observed, with the increase in network width and depth, the errors (2.11) consistently decrease, allowing for a better approximation of the ground state. This aligns with the general understanding of neural networks,

Table 4: Error (2.11) of the DNNs with a shift layer with different architecture.

	$W=10$	$W=50$	$W=70$
$L=1$	3.15E-2	6.42E-2	4.33E-2
$L=2$	9.05E-3	4.46E-3	6.76E-3
$L=3$	8.16E-3	1.24E-3	5.12E-3
$L=4$	8.95E-3	1.63E-3	1.59E-3

which suggests that wider and deeper networks possess greater expressive capability, resulting in improved model performance and approximation ability [23, 41].

Let ϕ_L^W represent the DNN (3.1) with width W and depth L. We consider the following cases: ϕ_1^{10}, ϕ_1^{70}, ϕ_4^{10} and ϕ_4^{70}. Their profiles are shown in Fig. 3(a). It can be observed that when the DNN is not wide enough or deep enough, the DNN (3.1) does not learn the details (at the corners of the ground state) effectively. Fig. 3(b) clearly shows that the pointwise errors of ϕ_4^{70} is significantly lower, and it achieves a lower relative error and a reduced action functional value with fewer training steps, as shown in Figs. 3(c) and (d). These results indicate that when the network architecture is increased to $L=4$ and $W=70$, the DNN approach of ϕ_4^{70} exhibits both high accuracy and efficiency.

Remark 3.1. Notably, our method and the Deep Ritz method [23] share some similarity. They both solve elliptic equations by transforming them into some optimization problems. However, we remark here that they are in fact essentially different. The elliptic equation that can be addressed by Deep Ritz possesses a unique solution, which according to Ritz theorem, is completely equivalent to finding the global minimum of the associated variational problem. In contrast, Eq. (1.2) that we aim to solve has infinitely many solutions, and the action ground state is a special one. Moreover, the action ground state solution is not unique either. It is determined up to a phase shift. Therefore, the proposed DNN approach is not a direct Deep Ritz method.

4 DNN approach based on the L^{p+1}-normalization formulation

In this section, we present an L^{p+1}-normalized DNN to approximate the ground state based on the L^{p+1}-normalized formulation (2.3). Compared to the methods of unconstrained formulation, prior research on solving action ground states with traditional numerical methods have indicated that constrained approaches can sometimes

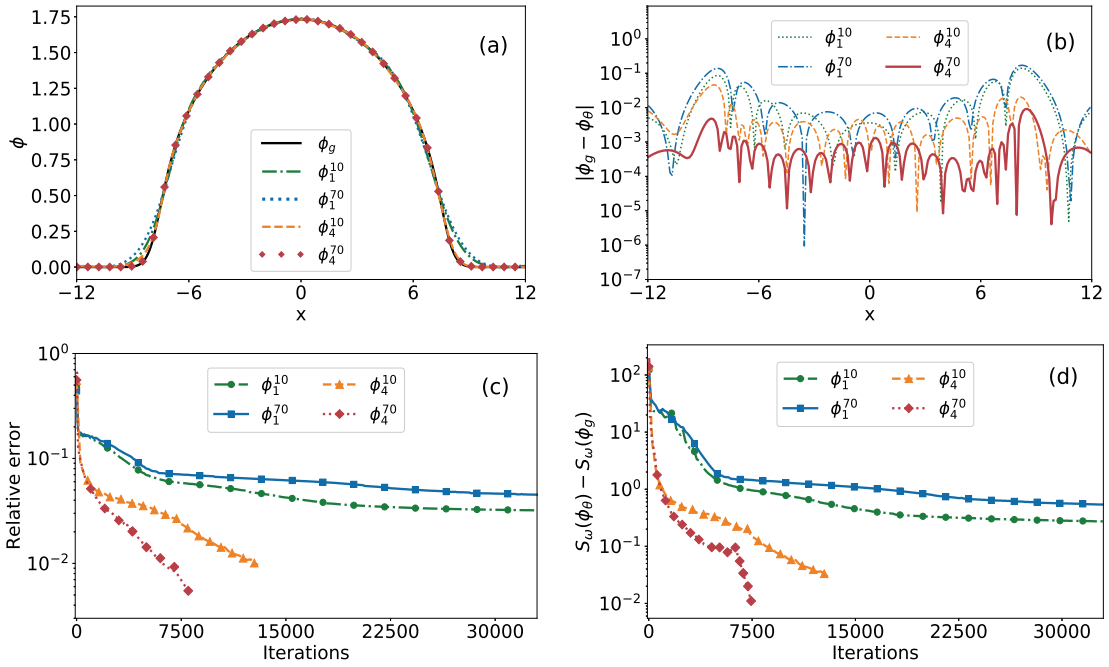

Figure 3: Some DNNs (3.1) with different architecture for Example 3.1: (a) the profiles of exact solution and numerical solutions; (b) pointwise errors; (c,d) the change of error (2.11) and $S_\omega(\phi_\theta)-S_\omega(\phi_g)$ during the iterations (in logarithmic scale).

exhibit higher computational efficiency [38]. Therefore, developing effective DNN approaches for these constrained formulations is of significant importance.

4.1 L^{p+1}-normalized DNN

As outlined in Section 2, once the minimizer u_* of the quadratic functional $Q(u)$ (2.2) on the L^{p+1} unit sphere $\mathcal{S}_{p+1}$ is determined, the ground state can be obtained through the transformation given by (2.4). Therefore, the ground state problem is transformed into solving (2.3) to obtain the minimizer u_*. To accomplish this, we design an **L^{p+1}-normalized deep neural network (L^{p+1}-norm DNN)**, which is expressed in the following form:

$$u_{\mathcal{L}}(\mathbf{x};\theta)=\mathcal{L}\circ\mathcal{T}\circ\mathbf{F}_{L+1}\circ\sigma\circ\mathbf{F}_L\circ\sigma\circ\cdots\circ\mathbf{F}_2\circ\sigma\circ\mathbf{F}_1(\mathbf{x}),\quad \mathbf{x}\in\mathbb{R}^d, \tag{4.1}$$

where the *L^{p+1}-normalization layer* $\mathcal{L}$ is defined as:

$$\mathcal{L}(\mathbf{y}):=\frac{\mathbf{y}}{\|\mathbf{y}\|_{L^{p+1}}},\quad \mathbf{y}=\mathbf{y}(\mathbf{x}):\mathbb{R}^d\to\mathbb{R}^{n_{L+1}}. \tag{4.2}$$

Then the output of L^{p+1}-norm DNN (4.1) satisfies $u_{\mathcal{L}} \in \mathcal{S}_{p+1}$. The shift layer $\mathcal{T}$ is defined as (3.2), and the activation function in (4.1) is chosen to be $\sigma = \tanh$.

Naturally, the quadratic functional Q defined by (2.2) is utilized as the loss function to train the L^{p+1}-norm DNN (4.1) and optimize its parameters, i.e., Loss:= $Q(u_{\mathcal{L}})$. After truncating the computational domain as U and selecting integration points $\{\mathbf{x}_j\}_{j=1}^N \subset U$, we have

$$\text{Loss}(\theta) = \frac{|U|}{N} \sum_{j=1}^{N} \left[\frac{1}{2} |\nabla u_{\mathcal{L}}(\mathbf{x}_j;\theta)|^2 + V(\mathbf{x}_j) |u_{\mathcal{L}}(\mathbf{x}_j;\theta)|^2 + \omega |u_{\mathcal{L}}(\mathbf{x}_j;\theta)|^2 \right]. \tag{4.3}$$

Thus, the constrained optimization problem (2.3) defined in the function space $\mathcal{S}_{p+1}$ is transformed into an unconstrained minimization (2.9) in finite dimensional parameter space, where the loss function Loss(θ) in (2.9) is defined by (4.3). It is important to note that $u_{\mathcal{L}}$ approximates not the ground state ϕ_g but rather the minimizer u_* of the quadratic minimization problem (2.3). The approximation of the ground state represented as $\phi_{\mathcal{L}}$ can be obtained through the following transformation:

$$\phi_{\mathcal{L}}(\mathbf{x}) := \left(\frac{Q(u_{\mathcal{L}})}{-\beta} \right)^{\frac{1}{p-1}} u_{\mathcal{L}}(\mathbf{x}). \tag{4.4}$$

4.2 Numerical experiment

We will demonstrate the superiority of the normalization layer (4.2) in addressing the constraint in 1D case, followed by an investigation into the impact of network architecture on the approximation performance of the L^{p+1}-norm DNN (4.1). The constrained problem (2.3) by L^{p+1}-norm DNN now takes the form:

$$\theta^* = \arg\min_{\theta} \frac{2I}{N} \sum_{j=1}^{N} \left[\frac{1}{2} |\nabla u_{\mathcal{L}}(x_j;\theta)|^2 + V(x_j) |u_{\mathcal{L}}(x_j;\theta)|^2 + \omega |u_{\mathcal{L}}(x_j;\theta)|^2 \right].$$

4.2.1 Superiority of L^{p+1}-normalization layer

In the L^{p+1}-norm DNN (4.1), we enforce the L^{p+1}-normalization as a hard constraint into the network, thereby introducing the L^{p+1}-normalization layer $\mathcal{L}$. To demonstrate the superiority of this approach in handling the constraint, we compare it with the common soft-constraint way. In detail, we apply the L^{p+1}-normalization constraint as a penalty term in the loss function to encourage the network's output satisfy $u_{\mathcal{S}} \in \mathcal{S}_{p+1}$, and the corresponding loss function is defined as

$$\text{Loss}(\theta) := Q(u_{\mathcal{S}}) + \eta \cdot \left(\|u_{\mathcal{S}}\|_{L^{p+1}} - 1 \right)^2,$$

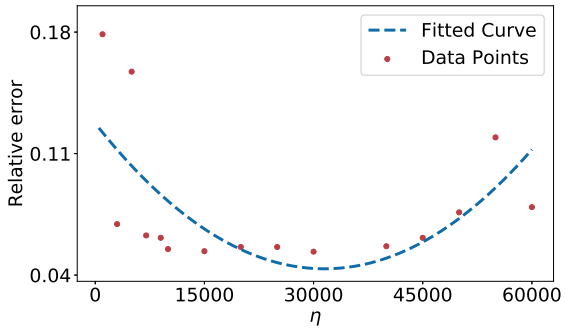

Figure 4: The variation of the relative error (2.11) of $\phi_{\mathcal{S}}$ with respect to the penalty factor η.

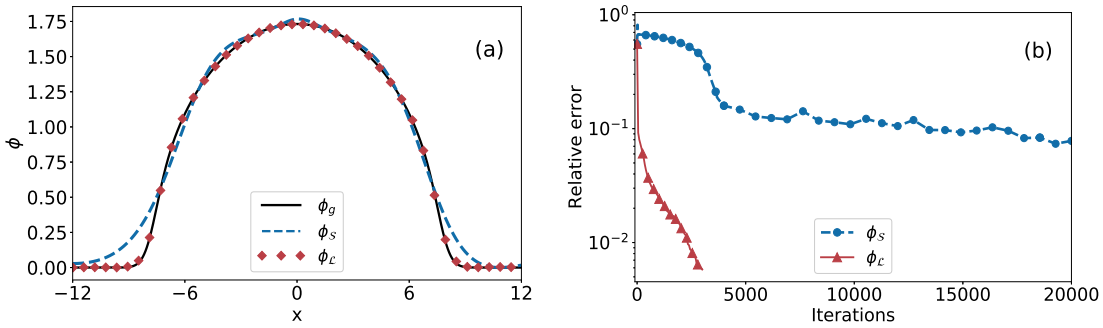

Figure 5: Results for Example 3.1 with $\phi_{\mathcal{L}}$ and $\phi_{\mathcal{S}}$: (a) the profiles of exact solution and numerical solutions; (b) the change of relative error (2.11) during the iterations.

where η denotes the penalty factor, $u_{\mathcal{S}}$ represents the DNN without the normalization layer $\mathcal{L}$ in (4.1), and the corresponding ground state approximate solution given by transformation (4.4) is denoted as $\phi_{\mathcal{S}}$. The second case is the L^{p+1}-norm DNN that minimizes the loss function (4.3), yielding the second approximation of the ground state $\phi_{\mathcal{L}}$. The hyper-parameters are set as $L=4$, $W=50$, $tol=10^{-6}$, $N=128$ and the Gaussian pre-training is applied for 1000 iterations for the two cases to fit the function $u_0=\mathrm{e}^{-x^2/10}/(5\pi/2)^{1/8}\in\mathcal{S}_{p+1}$. To ensure a fair comparison, the optimal penalty factor for $\phi_{\mathcal{S}}$ should be first determined. Fig. 4 illustrates how the relative error (2.11) of $\phi_{\mathcal{S}}$ varies with different values of the penalty factor η. By fitting this relationship with a quadratic curve, we identify η=31490 as the optimal value, corresponding to the minimum point of the curve. Then we compare the performance of $\phi_{\mathcal{S}}$ and $\phi_{\mathcal{L}}$.

Table 5 presents the number of iterations, computation time, and relative errors of the DNN $\phi_{\mathcal{S}}$ and the L^{p+1}-norm DNN $\phi_{\mathcal{L}}$. The results clearly indicate that

Table 5: Number of iterations, computational time and error (2.11) for $\phi_{\mathcal{S}}$ and $\phi_{\mathcal{L}}$.

	Iterations	Time	Error (2.11)
$\phi_{\mathcal{S}}$	23000	124s	6.84E-2
$\phi_{\mathcal{L}}$	3400	22s	4.51E-3

the $\phi_{\mathcal{L}}$, generated by the L^{p+1}-norm DNN (4.1) with the L^{p+1}-normalization layer $\mathcal{L}$, achieves a better accuracy with fewer iterations and shorter computation time. Fig. 5(a) shows that $\phi_{\mathcal{S}}$, which is produced by the DNN without the normalization layer, deviates significantly from the exact solution at the corners and has an overall profile that is poorly learned. In contrast, $\phi_{\mathcal{L}}$ captures most information of the exact solution ϕ_g. Furthermore, as shown in Fig. 5(b), the error (2.11) associated with $\phi_{\mathcal{L}}$ decreases at a more rapid rate. The comparative results above indicate that, for solving the action ground state problem under the L^{p+1}-normalization formulation, our proposed L^{p+1}-norm DNN with a hard constraint outperforms the DNN with a soft penalty constraint.

4.2.2 Influence of network architecture

Then we we conduct a systematic numerical investigation into the influence of the network architecture on L^{p+1}-norm DNN (3.1), with the hyper-parameters set as $L\in\{1,2,3,4\}$, $W\in\{10,50,70\}$, $tol=10^{-7}$, $N=256$. We apply the L^{p+1}-norm DNN approach to solve Example 3.1 and present the errors (2.11) of $\phi_{\mathcal{L}}$ obtained by networks with different depths and widths in Table 6. It can be observed that as the width and depth increase, the error consistently decreases, which indicates that wider and deeper L^{p+1}-norm DNNs can achieve a better approximation of the ground state. Setting ϕ_L^W as the L^{p+1}-norm DNN (4.1) with width W and depth L, we consider the following four cases: ϕ_1^{10}, ϕ_1^{70}, ϕ_4^{10} and ϕ_4^{70}. Fig. 6 illustrates the exact and numerical solutions, pointwise errors, the change of relative errors and $S_\omega(\phi_\theta)-S_\omega(\phi_g)$ along with iterations obtained from these four different network architectures. It can be seen that ϕ_4^{70} provides the best approximation of the ground state, with the minimum pointwise error, and achieves optimal accuracy and efficiency. This pattern aligns with the general understanding of neural networks.

5 DNN approach based on the Nehari constrained formulation

In this section, we focus on the Nehari constrained formulation (2.5), introducing a projection layer to handle the constraint, thereby developing a Nehari projected

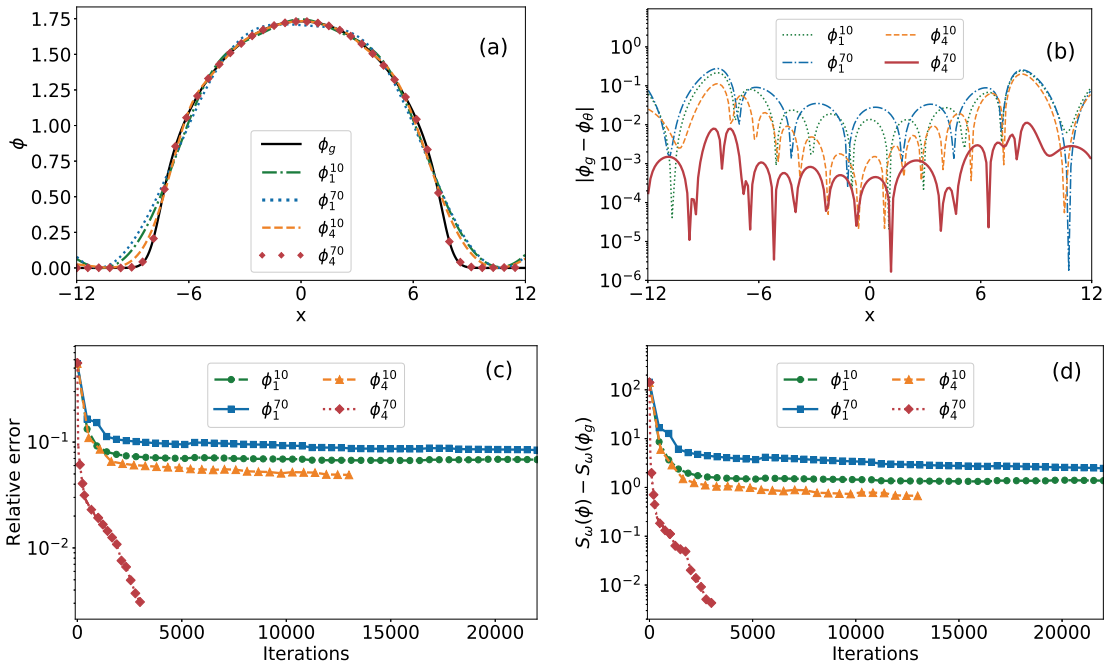

Figure 6: Some L^{p+1}-norm DNNs with different architecture for Example 3.1: (a) the profiles of exact solution and numerical solutions; (b) pointwise errors; (c,d) the change of error (2.11) and $S_\omega(\phi_\theta)-S_\omega(\phi_g)$ during the iterations (in logarithmic scale).

Table 6: Error (2.11) of L^{p+1}-norm DNNs with different architecture.

	$W=10$	$W=50$	$W=70$
$L=1$	6.78E-2	6.86E-2	8.18E-2
$L=2$	6.36E-2	4.40E-2	6.98E-3
$L=3$	6.37E-2	7.17E-3	3.87E-3
$L=4$	4.55E-2	4.05E-3	2.49E-3

deep neural network to compute the action ground state. Numerical experiments will demonstrate the effectiveness of this network in approximating the action ground state.

5.1 Nehari projected DNN

The Nehari constrained formulation (2.5) requires seeking the minimizer of the action functional $S_\omega(\phi)$ over the Nehari manifold $\mathcal{N}_\omega$. According to Proposition 2.1, any function $\phi\in X\backslash\{0\}$ can be projected onto the Nehari manifold, meaning that $\sigma_\omega(\phi)\phi\in\mathcal{N}_\omega$, where the projection coefficient $\sigma_\omega(\phi)$ is defined by (2.6). Moreover,

drawing inspiration from the superiority of the L^{p+1}-normalization layer, we propose adding a projection layer into the network to perform this operation. Adopting this strategy leads to the establishment of a **Nehari projected deep neural network (Nehari-proj DNN)**, which is expressed in the following specific form:

$$\phi_{\mathcal{N}}(\mathbf{x};\theta)=\mathcal{N}\circ\mathcal{T}\circ\mathbf{F}_{L+1}\circ\sigma\circ\mathbf{F}_{L}\circ\sigma\circ\cdots\circ\mathbf{F}_{2}\circ\sigma\circ\mathbf{F}_{1}(\mathbf{x}),\quad \mathbf{x}\in\mathbb{R}^{d},\tag{5.1}$$

where $\mathcal{T}$ denotes the shift layer defined by (3.2), $\mathcal{N}$ represents the *Nehari projection layer*, expressed as $\mathcal{N}(\mathbf{y}):=\sigma_{\omega}(\mathbf{y})\mathbf{y}$. The projection coefficient $\sigma_{\omega}(\mathbf{y})$ is defined as:

$$\sigma_{\omega}(\mathbf{y})=\left[\frac{\|\nabla\mathbf{y}\|_{L^2}^2+2\int_{\mathbb{R}^d}V|\mathbf{y}|^2d\mathbf{x}+2\omega\|\mathbf{y}\|_{L^2}^2}{-2\beta\|\mathbf{y}\|_{L^{p+1}}^{p+1}}\right]^{\frac{1}{p-1}},\tag{5.2}$$

where $\mathbf{y}=\mathbf{y}(\mathbf{x}):\mathbb{R}^d\to\mathbb{R}^{n_{L+1}}$. The Nehari projection layer $\mathcal{N}$ ensures the output of Nehari-proj DNN satisfies $K_{\omega}(\phi_{\mathcal{N}}(\mathbf{x};\theta))=0$, thus $\phi_{\mathcal{N}}(\mathbf{x};\theta)\in\mathcal{N}_{\omega}$. Although the projection coefficient (5.2) contains a gradient, the capability of DNNs to compute gradients via automatic differentiation and the chain rule makes the calculation of (5.2) is easy. Once the DNN is defined, we need to specify the loss function. The minimization objective in Nehari constrained formulation (2.5) is the action functional (1.3), thus we use the action functional S_{ω} to define the loss function, as in (3.3). Consequently, the ground state problem under the Nehari manifold constraint (2.5) is transformed into the parameter minimization problem (2.9) through the Nehari-proj DNN.

5.2 Numerical experiment

We utilize the Nehari-proj DNN $\phi_{\mathcal{N}}$ (4.1) to solve for the ground state in the 1D Example 3.1, with the hyper-parameters set to $L\in\{1,2,3,4\}$, $W\in\{10,50,70\}$, $tol=10^{-7}$, $N=256$. Gaussian pre-training is conducted for 1000 iterations to obtain the suitable initial parameters θ^0, and the initial function ϕ_0 should satisfy the Nehari manifold constraint . Specifically, we pre-train the Nehari-proj DNN by the projected function $\phi_0=\sigma_{\omega}(\tilde{\phi}_0)\tilde{\phi}_0\in\mathcal{N}_{\omega}$, where $\tilde{\phi}_0=\mathrm{e}^{-x^2/10}$ is a Gaussian function and $\sigma_{\omega}(\cdot)$ is defined in (2.6).

Table 7 displays the errors (2.11) recorded for Nehari-proj DNNs with varying depths and widths, where the minimum error in Table 7 can reduce to 1.38×10^{-3}, indicating that the Nehari-proj DNN provides a valid approximation. Moreover, as the network width and depth increase, a consistent reduction in errors is observed, facilitating a more accurate approximation of the ground state. Setting ϕ_L^W as the Nehari-proj DNN (5.1) with width W and depth L, we present the exact and numerical solutions, pointwise errors and the change of error (2.11) and $S_{\omega}(\phi_{\theta})-$

Table 7: Error (2.11) of Nehari-proj DNNs with different architecture.

	$W=10$	$W=50$	$W=70$
$L=1$	1.04E-1	6.23E-2	1.46E-1
$L=2$	5.41E-2	2.93E-2	2.90E-2
$L=3$	3.34E-2	2.97E-2	3.95E-2
$L=4$	5.74E-3	1.95E-3	1.38E-3

$S_\omega(\phi_g)$ along with iterations for ϕ_1^{10}, ϕ_1^{70}, ϕ_4^{10}, and ϕ_4^{70} in Fig. 7. From the profiles of the numerical solutions provided in Fig. 7(a), it can be seen that ϕ_4^{70} gives the best approximation, while the approximations of the other three DNNs are quite poor. From Figs. 7(b) and (c), it can also be observed that ϕ_4^{70} has the smallest pointwise error and the highest accuracy. The optimal efficiency of ϕ_4^{70} is demonstrated in Fig. 7(d). The numerical results above demonstrate that the Nehari-proj DNN can effectively solve the action ground state problem under the Nehari constrained formulation, and appropriately increasing the network architecture can yield better approximations of the ground state.

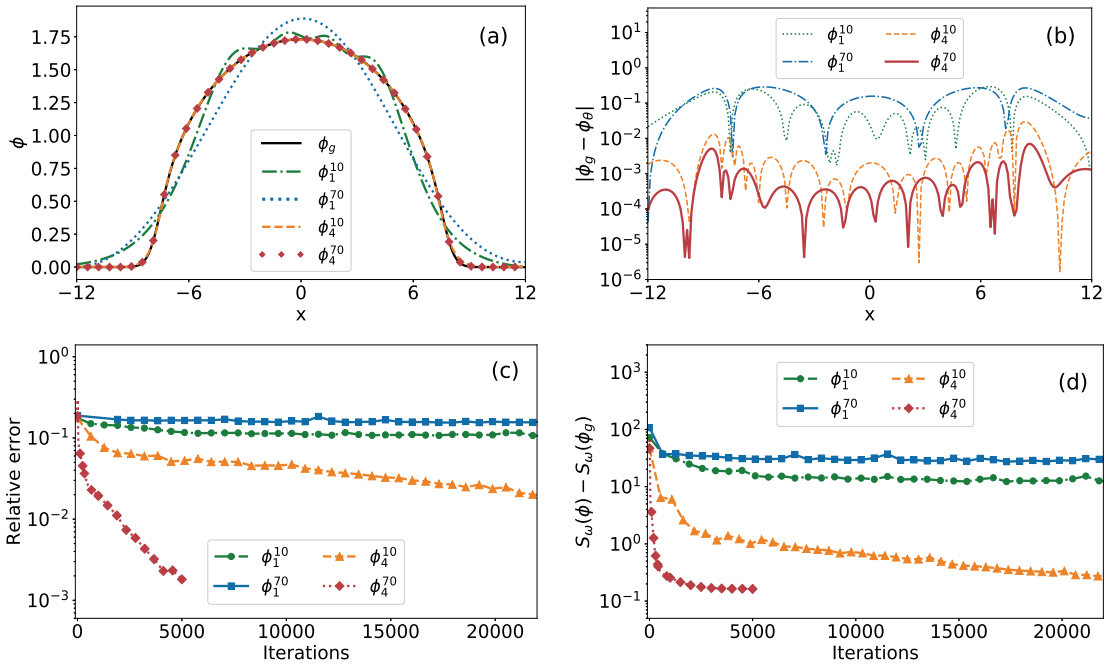

Figure 7: Some Nehari-proj DNNs with different architecture for Example 3.1: (a) the profiles of exact solution and numerical solution; (b) pointwise errors; (c,d) the change of error (2.11) and $S_\omega(\phi_\theta) - S_\omega(\phi_g)$ during the iterations (in logarithmic scale).

6 Applications and extensions

In this section, we will compare the performance of the proposed deep neural network (DNN) approaches and extend the DNN appraoch to address the harmonic-plus-optical lattice potential and 2D cases. Additionally, we will discuss improvement in both accuracy and efficiency.

6.1 Comparison of the DNN approaches

The three previously proposed DNN approaches are designed based solely on different formulations of the action ground state, and theoretical analysis alone can not clearly determine which method performs better. Therefore, we will conduct numerical experiments to compare their efficiency.

We use the three proposed DNN approaches to solve the Example 3.1 and set the hyper-parameters as $L=4$, $W=70$, $N=256$. We apply the Gaussian pre-training for 1000 iterations to fit a Gaussian function $\phi_0=\mathrm{e}^{-x^2/10}$, and then train the DNN with a shift layer (3.1), the L^{p+1}-norm DNN (4.1), and the Nehari-proj DNN (5.1) to approximate the ground state solution. To ensure a fair comparison, a unified criterion for stopping the training is established as follows:

$$\left|\sum_{k=100(j+1)}^{100(j+2)} S_\omega(\phi_{\theta^k})-\sum_{k=100j}^{100(j+1)} S_\omega(\phi_{\theta^k})\right| \Bigg/ \left|\sum_{k=100j}^{100(j+1)} S_\omega(\phi_{\theta^k})\right| <10^{-7}, \quad j=0,1,2\cdots,$$

where the ϕ_{θ^k} represents the DNN approximation obtained at the k-th training step. Table 8 presents the number of iterations, computation time and relative error (2.11) of these DNN approaches under different formulations. It can be seen that the L^{p+1}-norm DNN approach is the most efficient, requiring the fewest training iterations and taking only about half the computational time of the other two methods. Fig. 8(c) further shows how the relative error (2.11) of the three DNN approaches varies with computation time during the calculation process. It can be observed that the L^{p+1}-norm DNN reduces the error (2.11) the fastest, making it the best one.

Table 8: Comparison of the DNN approaches: DNN with a shift layer $\phi_{\mathcal{T}}$; $\phi_{\mathcal{L}}$ given by L^{p+1}-norm DNN; Nehari-proj DNN $\phi_{\mathcal{N}}$.

	Iterations	Time	Error (2.11)
$\phi_{\mathcal{T}}$	13500	72s	1.59E-3
$\phi_{\mathcal{L}}$	3500	28s	2.42E-3
$\phi_{\mathcal{N}}$	5500	82s	1.38E-3

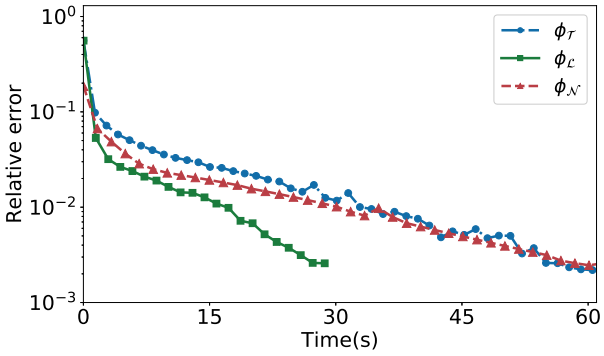

Figure 8: Error (2.11) during computational time for different DNN approaches: DNN with a shift layer $\phi_{\mathcal{T}}$; $\phi_{\mathcal{L}}$ given by L^{p+1}-norm DNN; Nehari-proj DNN $\phi_{\mathcal{N}}$.

The conclusion from the above comparison guides us to apply the L^{p+1}-normalized DNN approach for solving the ground state in the subsequent numerical experiments.

6.2 Application to harmonic-plus-optical lattice potential

The harmonic-plus-optical lattice potential proposed in nonlinear optics research consists of a harmonic term and a periodic optical lattice term, where the triangular terms of the latter incorporate some high-frequency information [29]. When utilizing the DNN approach to address the ground state problem defined by this potential function, we will clarify the role of the activation function in influencing the approximation performance of the DNN, thereby emphasizing the importance of selecting an appropriate activation function for specific problems. To this end, a 1D example concerning the harmonic-plus-optical lattice potential will be examined.

Example 6.1. Take d=1, p=3, β=10, ω=−30 in NLS (1.1), employ the harmonic-plus-optical lattice potential $V(x)=\frac{1}{2}x^2+25\sin^2(\frac{\pi x}{3})$, and fix the computational domain U=(−12,12). Here, the exact action for the ground state is $S_\omega(\phi_g)$=−136.7589, which can be obtained using the GF-BF method.

The L^{p+1}-norm DNN (4.1) is utilized to solve Example 6.1, with hyper-parameters set to $W=10$, $L\in\{2,3,4\}$, $tol=10^{-5}$, $N=128$ and the activation functions 'tanh' and 'sin' are employed to the L^{p+1}-norm DNN (4.1). Table 9 and Fig. 9 present the performance of the L^{p+1}-norm DNNs with two different activation functions. As illustrated in Fig. 9(a), the L^{p+1}-norm DNN defined by the activation function 'tanh' only captures three wave structures of the ground state but fails to learn the two wave structures at the edges, indicating a significant limitation in capturing local features of high-frequency fluctuations. Table 9 further shows that the

Table 9: Number of iterations, computational time and error (2.11) of L^{p+1}-norm DNN (4.1) with 'tanh' or 'sin' as the activation function.

σ	$W\times L$	Iterations	Time	Error (2.11)
tanh	10×2	46200	331s	3.20E-1
	10×3	8500	47s	3.59E-1
	10×4	4800	28s	3.75E-1
sin	10×2	8900	43s	5.38E-2
	10×3	6100	33s	3.19E-2
	10×4	2800	16s	2.76E-2

corresponding error (2.11) consistently remains on the order of 10^{-1}. In contrast, the L^{p+1}-norm DNN defined by activation function 'sin' can fully capture the local oscillations of the ground state, with the error reduced to 10^{-2}, demonstrating a superior approximation performance. Moreover, the comparison of iterations and computational time presented in Table 9 indicates that the L^{p+1}-norm DNN defined by 'sin' requires significantly less computational cost, allowing the action functional to decrease at a faster rate to lower levels, as shown in Fig. 9(d). We conclude that, compared to activation function 'tanh', 'sin' is more suitable for use in L^{p+1}-norm DNN to approximate the action ground state influenced by optical potential.

6.3 Application to two-dimensional problem

We now consider the 2D case, i.e., $d=2$ and $\mathbf{x}=(x,y)$ in NLS (1.1). The straightforward extension of the L^{p+1}-norm DNN approach to 2D case then considers the optimization:

$$\theta^*=\arg\min_{\theta}\frac{|U|}{N_x\times N_y}\sum_{j=1}^{N_x}\sum_{k=1}^{N_y}\Big[\frac{1}{2}|\nabla u_{\mathcal{L}}(x_j,y_k;\theta)|^2+V(x_j,y_k)|u_{\mathcal{L}}(x_j,y_k;\theta)|^2 +\omega|u_{\mathcal{L}}(x_j,y_k;\theta)|^2\Big], \tag{6.1}$$

with

$$u_{\mathcal{L}}(x,y;\theta)=\mathcal{L}\circ\mathcal{T}\circ\mathbf{F}_{L+1}\circ\sigma\circ\mathbf{F}_L\circ\sigma\circ\cdots\circ\mathbf{F}_2\circ\sigma\circ\mathbf{F}_1(x,y). \tag{6.2}$$

Now a 2D example will be examined.

Example 6.2. Take $d=2$, $p=3$, $\beta=400$, $\omega=-10$ in NLS (1.1), and employ the harmonic oscillator potential $V(x,y)=\frac{1}{2}(x^2+y^2)$. The computational domain is fixed as $U=(-6,6)\times(-6,6)$. Set the hyper-parameters as $L\in\{1,2,3,4\}$, $W\in\{10,50,70\}$,

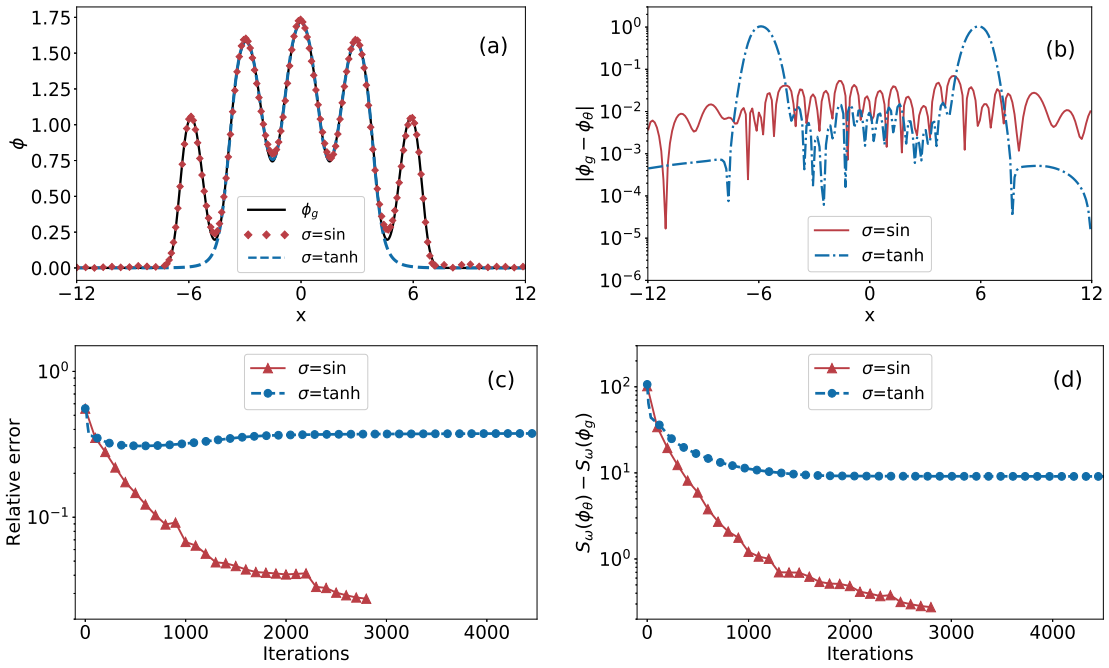

Figure 9: L^{p+1}-norm DNNs with different activation functions for Example 6.1 with $L=4$, $W=10$: (a) the profiles of exact solution and numerical solutions; (b) pointwise errors; (c), (d) the change of error (2.11) and $S_\omega(\phi_\theta)-S_\omega(\phi_g)$ during the iterations (in logarithmic scale).

$tol=10^{-7}$ and $N_x=N_y=64$. The activation function $\sigma=\tanh$ is used in L^{p+1}-norm DNN (4.1). Here, the exact action for the ground state is $S_\omega(\phi_g)=-2.5234$.

Table 10 presents the relative errors (2.11) of L^{p+1}-norm DNN (6.2) with varying widths and depths. It is evident that as the network structure expands, the error can be reduced to 1.67×10^{-2}. Analogous to the 1D case, wider and deeper networks yield superior approximation results. Under the configuration of $L=4$, $W=70$, we show the profiles of the solutions and corresponding pointwise errors in Fig. 10. It can be observed that the L^{p+1}-norm DNN offers valid approximation, with the pointwise errors remaining consistently below 1%.

Next, we solve the ground state of the NLS (1.1) defined by the harmonic-plus-optical lattice potential in 2D case.

Example 6.3. Take $d=2$, $p=3$, $\beta=400$, $\omega=-10$ in NLS (1.1), and employ the harmonic-plus-optical lattice potential

$$V(x,y)=\frac{1}{2}(x^2+y^2)+\frac{5}{2}\left(\sin^2\left(\frac{\pi x}{3}\right)+\sin^2\left(\frac{\pi y}{3}\right)\right).$$

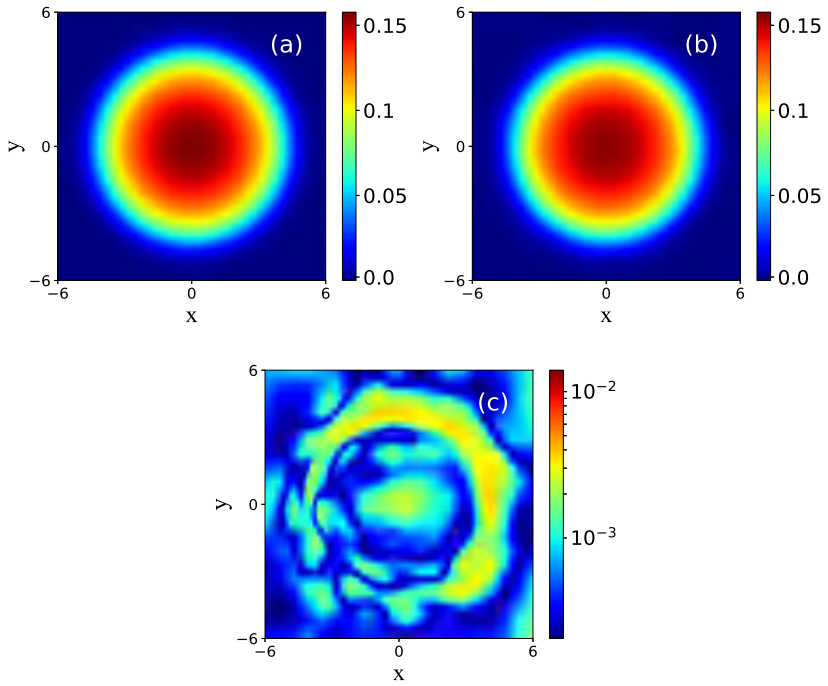

Figure 10: Results for Example 6.2 with $L=4$, $W=70$: (a) exact solution ϕ_g; (b) numerical solution of L^{p+1}-norm DNN; (c) pointwise errors.

The computational domain is fixed as $U=(-6,6)\times(-6,6)$. Setting the hyper-parameters as $L\in\{1,2,3,4\}$, $W\in\{10,50,70\}$, $tol=10^{-7}$ and $N_x=N_y=64$. The activation function $\sigma=\sin$ is used in (4.1). Here the exact action for the ground state is $S_\omega(\phi_g)=-1.0922$.

According to the conclusions regarding the activation function obtained in the 1D case, we choose 'sin' as the activate function for the harmonic-plus-optical lattice potential and apply the L^{p+1}-norm DNN (4.1) to solve Example 6.3. Table 11 presents the relative errors (2.11) of L^{p+1}-norm DNN with varying widths and depths

Table 10: Error (2.11) of L^{p+1}-norm DNNs with different architecture in Example 6.2.

	$W=10$	$W=50$	$W=70$
$L=1$	2.58E-1	1.86E-1	1.57E-1
$L=2$	1.22E-1	4.42E-2	3.52E-2
$L=3$	1.23E-1	2.12E-2	1.70E-2
$L=4$	3.95E-2	1.85E-2	1.67E-2

Table 11: Error (2.11) of L^{p+1}-norm DNNs with different architecture in Example 6.3.

	$W=10$	$W=50$	$W=70$
$L=1$	1.81E-1	1.87E-1	1.81E-1
$L=2$	1.88E-1	1.15E-1	1.44E-1
$L=3$	1.62E-1	7.53E-2	6.09E-2
$L=4$	1.16E-1	5.21E-2	3.69E-2

in Example 6.3 and Fig. 11 presents the numerical results for $L=4$, $W=70$, which shows that the activation function 'sin' continues to effectively enable the L^{p+1}-norm DNN to approximate the action ground state in the 2D case exactly.

The above results show that the proposed DNN approach can work well for 2D problems and the techniques we developed in 1D case can be straightforwardly extended to the 2D case. However, one more issue encountered in the high-dimensional case is the grid points used to approximate the action functional, e.g., the (x_j,y_k) in (6.1). Equal partitioning is not feasible in very high dimensions, so the common approach is to use random sampling instead. To demonstrate the capacity of the

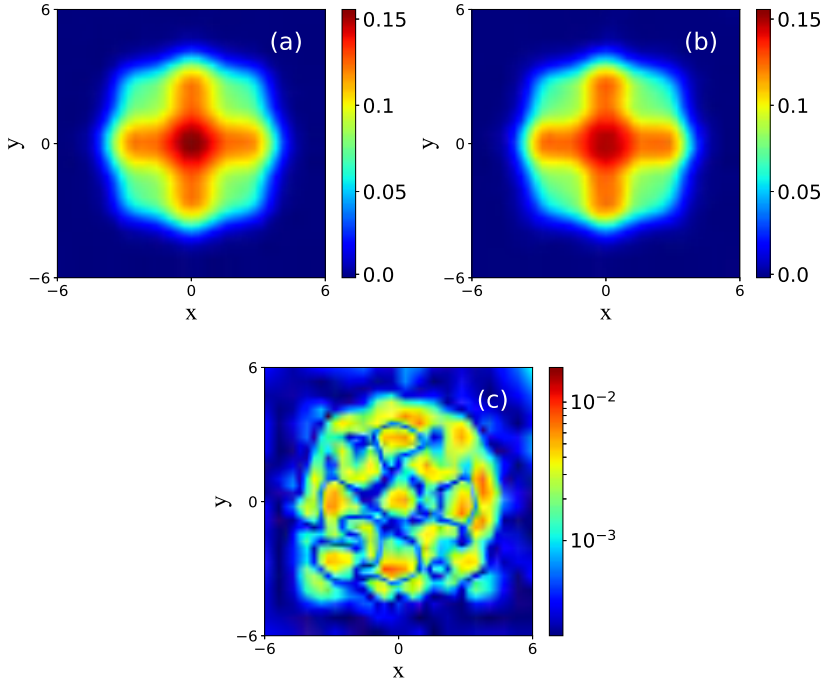

Figure 11: Results for Example 6.3 with $L=4$, $W=70$: (a) exact solution ϕ_g; (b) numerical solution of L^{p+1}-norm DNN; (c) pointwise errors.

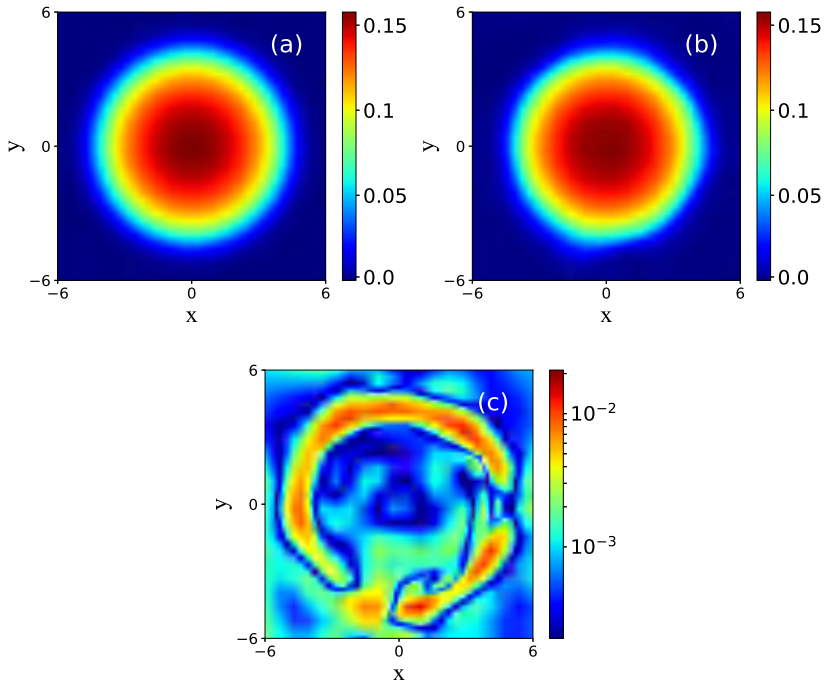

Figure 12: Results for Example 6.2 with $L=4$, $W=70$: (a) exact solution ϕ_g; (b) numerical solution of L^{p+1}-norm DNN with $N_b=1000$ randomly generated training points; (c) pointwise errors.

proposed DNN approach with random sampling, we consider the 2D ground state problem in Example 6.2. Now, we randomly generate some points according to the uniform distribution in computational domain U. The total number of the generated points is denoted as $N_b=1000$, which is much less than that used in Example 6.2 to train the DNN, i.e., $N_b<64\times64=4096$. The other setup remains the same as before, and we train the L^{p+1}-norm DNN with $L=4, W=70$.

Fig. 12 illustrates the profiles of the solutions and corresponding pointwise errors for $N_b=1000$. Obviously, the L^{p+1}-norm DNN continues to perform exceptionally well in capturing the action ground state. In addition, we compare the number of iterations, computation time and error (2.11) of the fixed uniform grid points

Table 12: Number of iterations, computational time and error (2.11) for L^{p+1}-norm DNNs with randomly selected training points and fixed equally distributed training points.

	Iterations	Time	Error (2.11)
fixed uniform grid points	37100	464.1s	1.67E-2
randomly generated grid points	40300	412.6s	3.37E-2

and random points in Table 12. As we can see, even with random points, L^{p+1}-norm DNN is still able to achieve accuracy comparable to that of uniform grid points. Furthermore, in terms of computational efficiency, although the number of iterations increases, the reduction in the size of the training set leads to a further decrease in both computation time and storage consumption.

6.4 Further improvement of accuracy and efficiency

In previous studies, our emphasis was on developing DNNs better suited for approximating the action ground state under different formulations, without conducting specific analyses on accuracy or efficiency. The subsequent discussion will illustrate that the accuracy and efficiency can be further improved. In the comparison of the three DNN approaches, the efficiency of the DNN with a shift layer (3.1) was inferior to that of the L^{p+1}-norm DNN (4.1). Here, we utilize the DNN with a shift layer (3.1) to solve the 1D Example 3.1 as a case study to discuss how to improve the accuracy and efficiency.

We begin with the improvement of accuracy. The best accuracy achieved in the previous 1D numerical experiments was only on the order of 10^{-3}. However, by utilizing deeper and wider networks, along with implementing stricter stopping criteria, the accuracy can be improved to 10^{-5}. The DNN with a shift layer (3.1) is employed to compute Example 3.1, with hyper-parameters set as $L\in\{9,13,15\}$, $W\in\{70,90,110,130\}$, $N=256$. Stricter stopping criteria were established to prevent premature termination of training. Specifically, training will stop when both of the following conditions are met: 1) the criterion for the change in the loss function defined by (2.10) is satisfied (with $tol=10^{-7}$); 2) the number of iterations exceeds 10^5. The errors corresponding to networks of varying widths and depths are presented in Table 13, indicating that the minimum error has been reduced to 2.17×10^{-5}. It is noteworthy that both the L^{p+1}-norm DNN and Nehari-proj DNN can also achieve comparable improvements in accuracy, reaching the order of 10^{-5}.

Next, the focus shifts to enhancing efficiency. Previously, the Adam optimizer was employed due to its widespread application in deep learning, adaptive learning rates, and stability. However, the selection of an optimizer is a critical consider-

Table 13: Error (2.11) of the DNN with a shift layer for different architecture in Example 3.1.

	$W=70$	$W=90$	$W=110$	$W=130$
$L=9$	3.59E-5	4.30E-5	2.50E-5	4.14E-5
$L=13$	2.01E-4	3.83E-5	3.57E-5	2.93E-5
$L=15$	2.23E-5	2.17E-5	4.78E-5	3.04E-5

Table 14: Number of iterations, computational time and error (2.11) for some DNNs with a shift layer (3.1) using Adam and L-BFGS.

	Iterations	Time	Error (2.11)
Adam	7200	41s	4.93E-3
L-BFGS	91	0.6s	3.20E-3

ation. Here, the L-BFGS optimizer is considered, recognized for its exceptional performance in high-precision optimization tasks [36]. The DNN with a shift layer (3.1) is utilized to compute Example 3.1, with the hyper-parameters set as $L=3$, $W=50$, $tol=10^{-6}$, $N=128$. In this context, the shift layer $\mathcal{T}$ originally defined by (3.2), is replaced with the activation function $\sigma=softplus$, which similarly ensures positive outputs, serving the same purpose as the shift layer. Table 14 presents the relevant data for the DNN with a shift layer (3.1) under both optimizers. It is evident that to achieve comparable error levels, the L-BFGS optimizer requires significantly fewer iterations and less computational time than Adam. However, the conclusion that the L-BFGS optimizer enhances computational efficiency cannot be extended to the other two constrained DNNs. Accelerating the convergence of DNNs is a crucial and intricate topic that will be further explored in future research.

7 Conclusions

In this paper, we proposed three different deep neural networks (DNNs) to solve the action ground state of the nonlinear Schrödinger equation based on three equivalent formulations of the problem. Firstly, for the unconstrained formulation of the action ground state, we proposed the DNN with a shift layer, which can efficiently produce the correct approximation of the ground state solution. For the other two constrained formulations, we implemented a normalization layer or a projection layer to the DNN to exactly satisfy the constraints, which lead to the proposed L^{p+1}-normalized DNN and the Nehari projected DNN. These proposed three DNNs provide effective unsupervised learning methods for computing the action ground state. Systematical numerical experiments have been conducted to demonstrate the effectiveness and superiority of the key techniques we proposed, including the introduction of the shift layer, Gaussian pre-training, the imposition of hard constraints in the network, and the selection of activation functions. Comparisons between the three DNN approaches were made, and some applications and extensions were further made. Future works will consider the applications of the proposed approaches to real high-dimensional problems and the physical parameter-generalizations.

Acknowledgements

The work is supported by the National Natural Science Foundation of China No. 1227 1413, the Natural Science Foundation of Hubei Province No. 2019CFA007 and the Fundamental Research Funds for the Central Universities No. 2042024kf0016.

References

[1] R. Altmann, P. Henning, and D. Peterseim, The J-method for the Gross-Pitaevskii eigenvalue problem, Numer. Math., 148 (2021), pp. 575–610.

[2] X. Antoine, A. Levitt, and Q. Tang, Efficient spectral computation of the stationary states of rotating Bose-Einstein condensates by preconditioned nonlinear conjugate gradient methods, J. Comput. Phys., 343 (2017), pp. 92–109.

[3] J. Arbunich, I. Nenciu, and C. Sparber, Stability and instability properties of rotating Bose-Einstein condensates, Lett. Math. Phys., 109 (2019), 1415–1432.

[4] A. H. Ardila, and H. Hajaiej, Global well-posedness, blow-up and stability of standing waves for supercritical NLS with rotation, J. Dyn. Differential Equations, 35 (2023), 1643–1665.

[5] X. Bai, and D. Zhang, Learning ground states of spin-orbit-coupled Bose-Einstein condensates by a theory-guided neural network, Phys. Rev. A, 104 (2021), 063316.

[6] T. A. Bakthavatchalam, S. Ramamoorthy, M. Sankarasubbu, R. Ramaswamy, and V. Sethuraman, Bayesian optimization of Bose-Einstein condensates, Sci. Rep., 11 (2021), 5054.

[7] W. Bao, and Y. Cai, Mathematical theory and numerical methods for Bose-Einstein condensation, Kinet. Relat. Models, 6 (2013), 1–135.

[8] W. Bao, Z. Chang, and X. Zhao, Computing ground states of Bose-Einstein condensation by normalized deep neural network, J. Comput. Phys., 520 (2025), 113486.

[9] W. Bao, and Q. Du, Computing the ground state solution of Bose-Einstein condensates by a normalized gradient flow, SIAM J. Sci. Comput., 25 (2004), 1674–1697.

[10] Y. Bengio, A. Courville, and P. Vincent, Representation learning: a review and new perspectives, IEEE Trans. Pattern Anal. Mach. Intell., 35 (2013), 1798–1828.

[11] H. Berestycki, and P. L. Lions, Nonlinear scalar field equations I: Existence of a ground state, Arch. Rat. Mech. Anal., 82 (1983), 313–345.

[12] H. Berestycki, and P. L. Lions, Nonlinear scalar field equations II: Existence of infinitely many solutions, Arch. Rat. Mech. Anal., 82 (1983), 347–375.

[13] M. Caliari, A. Ostermann, S. Rainer, and M. Thalhammer, A minimisation approach for computing the ground state of Gross-Pitaevskii systems, J. Comput. Phys., 228 (2009), 349–360.

[14] E. Cancès, R. Chakir, and Y. Maday, Numerical analysis of nonlinear eigenvalue problems, J. Sci. Comput., 45 (2010), 90–117.

[15] L. D. Carr, C. W. Clark, and W. P. Reinhardt, Stationary solutions of the one-dimensional nonlinear Schrödinger equation. I. Case of repulsive nonlinearity, Phys. Rev. A, 62 (2000), 063610.
[16] S. Chang, C. Chien, and B. Jeng, Computing waves of nonlinear Schrödinger equations: A time-independent approach, J. Comput. Phys., 226 (2007), 104–130.
[17] Z. Chang, J. Z. Yang, and X. Zhao, Solving initial-terminal value problem of time evolutions by a deep least action method: Newtonian dynamics and wave equations, Phys. Rev. E, 110 (2024), 045311.
[18] J. Chen, I. Chern, and W. Wang, Exploring ground states and excited states of spin-1 Bose-Einstein condensates by continuation methods, J. Comput. Phys., 230 (2011), 2222–2236.
[19] S. Cuccagna, Stabilization of solutions to nonlinear Schrödinger equations, Commun. Pure Appl. Math., 54 (2001), 1110–1145.
[20] I. Danaila, and P. Kazemi, A new Sobolev gradient method for direct minimization of the Gross-Pitaevskii energy with rotation, SIAM J. Sci. Comput., 32 (2010), 2447–2467.
[21] I. Danaila, and B. Protas, Computation of ground states of the Gross-Pitaevskii functional via Riemannian optimization, SIAM J. Sci. Comput., 39 (2017), B1102–B1129.
[22] S. Dovetta, E. Serra, and P. Tilli, Action versus energy ground states in nonlinear Schrödinger equations, Math. Ann., 385 (2023), 1545–1576.
[23] W. E, and B. Yu, The deep Ritz method: a deep learning-based numerical algorithm for solving variational problems, Commun. Math. Stat., 6 (2018), 1–12.
[24] R. Fukuizumi, Stability and instability of standing waves for the nonlinear Schrödinger equation with harmonic potential, Discrete Contin. Dyn. Syst., 7 (2001), 525–544.
[25] R. Fukuizumi, and M. Ohta, Instability of standing waves for nonlinear Schrödinger equations with potentials, Differ. Integral Equ. 16 (2003), 691–706.
[26] X. Glorot, and Y. Bengio, Understanding the difficulty of training deep feedforward neural networks, Proceedings of the 13th International Conference on Artificial Intelligence and Statistics, 9 (2010), 249–256.
[27] P. Henning, and D. Peterseim, Sobolev gradient flow for the Gross-Pitaevskii eigenvalue problem: global convergence and computational efficiency, SIAM J. Numer. Anal., 58 (2020), 1744–1772.
[28] G.E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, Improving neural networks by preventing co-adaptation of feature detectors, CoRR, arXiv:1207.0580, 2012.
[29] D. Jaksch, C. Bruder, J. I. Cirac, C. W. Gardiner, and P. Zoller, Cold bosonic atoms in optical lattices, Phys. Rev. Lett., 81 (1998), 3108–3111.
[30] L. Jeanjean, and S. Lu, On global minimizers for a mass constrained problem, Calc. Var. Partial Differential Equations, 64 (2022), 214.
[31] D. P. Kingma, and J. Ba, Adam: a method for stochastic optimization, ICLR,

(2015), 13.
[32] A. Krizhevsky, I. Sutskever, and G. E. Hinton, Imagenet classification with deep convolutional Neural Netw. Commun. ACM, 60 (2017), 84–90.
[33] Y. LeCun, Y. Bengio, and G. Hinton, Deep learning, Nature, 521 (2015), 436–444.
[34] X. Liang, H. Zhang, S. Liu, Y. Li, and Y. Zhang, Generation of Bose-Einstein condensates' ground state through machine learning, Sci. Rep., 8 (2018), 16337.
[35] W. Liu, and Y. Cai, Normalized gradient flow with Lagrange multiplier for computing ground states of Bose-Einstein condensates, SIAM J. Sci. Comput., 43 (2021), B219–B242.
[36] D. C. Liu, and J. Nocedal, On the limited memory BFGS method for large scale optimization, Math. Program., 45 (1989), 503–528.
[37] W. Liu, C. Wang, and X. Zhao, On action ground states of defocusing nonlinear Schrödinger equations, preprint, arXiv:2311.02890, 2023.
[38] W. Liu, Z. Wen, Y. Yuan, and X. Zhao, Computing defocusing action ground state of rotating nonlinear Schrödinger equation: methods via various formulations and comparison, preprint.
[39] W. Liu, Y. Yuan, and X. Zhao, Computing the action ground state for the rotating nonlinear Schrödinger equation, SIAM J. Sci. Comput., 45 (2023), A397–A426.
[40] R. L. Pego, and H. A. Warchall, Spectrally stable encapsulated vortices for nonlinear Schrödinger equations, J. Nonlinear Sci., 12 (2002), 347–394.
[41] M. Raissi, P. Perdikaris, and 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.
[42] J. Schmidhuber, Deep learning in neural networks: An overview, Neural Netw., 61 (2015), 85–117.
[43] B. I. Schneider, and D. L. Feder, Numerical approach to the ground and excited states of a Bose-Einstein condensed gas confined in a completely anisotropic trap, Phys. Rev. A, 59 (1999) 2232–2242.
[44] J. Shatah, and W. Strauss, Instability of nonlinear bound states, Commun. Math. Phys., 100 (1985), 173–190.
[45] K. Simonyan, and A. Zisserman, Very deep convolutional networks for large-scale image recognition, 3rd IAPR ACPR, (2015), 730–734.
[46] A. Soffer, and M. I. Weinstein, Multichannel nonlinear scattering for nonintegrable equations, Commun. Math. Phys., 133 (1990), 119–146.
[47] A. Soffer, and X. Zhao, Modulation equations approach for solving vortex and radiation in nonlinear Schrödinger equation, IMA J. Appl. Math., 83 (2018), 496–513.
[48] A. Soffer, and X. Zhao, A modulation equations approach for numerically solving the moving soliton and radiation solutions of NLS, Phys. D, 320 (2016), 77–88.
[49] A. Soffer, and X. Zhao, On multichannel solutions of nonlinear Schrödinger equations: algorithm, analysis and numerical explorations, J. Phys. A Math. Theory,

48 (2015), 135201.
[50] W. A. Strauss, Existence of solitary waves in higher dimensions, Commun. Math. Phys., 55 (1977), 149–162.
[51] C. Sulem, and P. L. Sulem, The Nonlinear Schrödinger Equation, Applied Sciences, 139, Springer, 1999.
[52] W. Sun, and Y. Yuan, Optimization Theory and Methods: Nonlinear Programming, Springer, 2006.
[53] C. Wang, Computing the least action ground state of the nonlinear Schrödinger equation by a normalized gradient flow, J. Comput. Phys., 471 (2022), 111675.
[54] M. I. Weinstein, Lyapunov stability of ground states of nonlinear dispersive evolution equations, Commun. Pure Appl. Math., 39 (1986), 51–67.
[55] X. Wu, Z. Wen, and W. Bao, A regularized Newton method for computing ground states of Bose-Einstein condensates, J. Sci. Comput., 73 (2017), 303–329.
[56] R. Zeng, and Y. Zhang, Efficiently computing vortex lattices in rapid rotating Bose–Einstein condensates, Comput. Phys. Commun., 180 (2009), 854–860.
[57] Q. Zhuang, and J. Shen, Efficient SAV approach for imaginary time gradient flows with applications to one- and multi-component Bose-Einstein condensates, J. Comput. Phys., 396 (2019) 72–88.