Journal of Computational Physics 520 (2025) 113486

Contents lists available at ScienceDirect

Journal of Computational Physics

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

# Computing ground states of Bose-Einstein condensation by normalized deep neural network

Weizhu Bao [a], Zhipeng Chang [b,*], Xiaofei Zhao [b,c]

[a] Department of Mathematics, National University of Singapore, 119076, Singapore
[b] School of Mathematics and Statistics, Wuhan University, 430072 Wuhan, China
[c] Computational Sciences Hubei Key Laboratory, Wuhan University, 430072 Wuhan, China

ARTICLE INFO

*Keywords:*
Bose-Einstein condensation
Gross-Pitaevskii equation
Normalized deep neural network
Ground state
Normalized layer
Mass normalization
Shift layer

ABSTRACT

We propose a normalized deep neural network (norm-DNN) for computing ground states of Bose-Einstein condensation (BEC) via the minimization of the Gross-Pitaevskii energy functional under unitary mass normalization. Compared with the traditional deep neural network for solving partial differential equations, two additional layers are added in training our norm-DNN for solving this kind of unitary constraint minimization problems: (i) a normalization layer is introduced to enforce the unitary mass normalization, and (ii) a shift layer is added to guide the training to non-negative ground state. The proposed norm-DNN gives rise to an efficient unsupervised approach for learning ground states of BEC. Systematical investigations are first carried out through extensive numerical experiments for computing ground states of BEC in one dimension. Extensions to high dimensions and multi-component are then studied in details. The results demonstrate the effectiveness and efficiency of norm-DNN for learning ground states of BEC. Finally, we extend the norm-DNN for computing the first excited states of BEC and discuss parameter generalization issues as well as compare with some existing machine learning methods for computing ground states of BEC in the literature.

## 1. Introduction

The idea of Bose-Einstein condensation (BEC) originated from A. Einstein's extension of S.N. Bose's work on quantum statistics of photons [15] to the case of Bose gases. A. Einstein predicted that particles in a Bose gas below a critical temperature would occupy the same quantum state, which was later experimentally realized [2] and considered as a new state of matter. Such state is now known as BEC and it has been widely concerned in atomic, molecule and optical (AMO) physics [9,47,48]. For a BEC, the fundamental physics is described by a many-body Schrödinger equation which is extremely difficult to solve due to the high dimensionality, and then researchers have been seeking for effective simplifications. For the dilute Bose gas below the critical temperature and confined in an external trap, by the mean-field approximation, the following Gross-Pitaevskii equation (GPE) (in dimensionless form) [9,27] is widely concerned in applications for modeling BEC:

$$i\frac{\partial\psi(\mathbf{x},t)}{\partial t} = -\frac{1}{2}\nabla^2\psi(\mathbf{x},t) + V(\mathbf{x})\psi(\mathbf{x},t) + \beta|\psi(\mathbf{x},t)|^2\psi(\mathbf{x},t), \quad \mathbf{x} = (x_1, x_2, \cdots, x_d)^\top \in \mathbb{R}^d, \tag{1.1}$$

* Corresponding author.
*E-mail addresses:* matbaowz@nus.edu.sg (W. Bao), changzhipeng@whu.edu.cn (Z. Chang), matzhxf@whu.edu.cn (X. Zhao).

https://doi.org/10.1016/j.jcp.2024.113486
Received 13 December 2023; Received in revised form 18 August 2024; Accepted 4 October 2024

Available online 9 October 2024

where $\psi := \psi(\mathbf{x}, t)$ represents the unknown complex-valued wave function and $d = 1, 2, 3$. Here $V(\mathbf{x})$ is a given real-valued function denoting the external trapping potential that in applications satisfies the confining condition

$$\lim_{|\mathbf{x}|\to\infty} V(\mathbf{x}) = \infty. \tag{1.2}$$

We would consider the following two types of potentials which are commonly used in experiments [9,47,48]:

$$\text{harmonic oscillator:} \quad V(\mathbf{x}) = \sum_{j=1}^{d} \frac{1}{2}\gamma_j^2 x_j^2; \quad \text{optical lattice:} \quad V(\mathbf{x}) = \sum_{j=1}^{d} \left(\frac{1}{2}\gamma_j^2 x_j^2 + a_j^2 \sin^2(\alpha_j x_j)\right); \tag{1.3}$$

with $\gamma_j, a_j, \alpha_j \in \mathbb{R}^+$. The parameter $\beta \in \mathbb{R}$ in (1.1) is a given constant with $\beta > 0$ denoting the repulsive interaction and $\beta < 0$ denoting the attractive case.

Though GPE is originated for BEC, as a mathematical model it is of interest in arbitrary dimension [41] and can be used to describe general nonlinear wave phenomenon. For the past few decades, extensive efforts have been devoted to solving the GPE for its dynamics or stationary solutions [3,9]. Along the dynamics in the model (1.1), the total mass and energy of the system are conserved, i.e.,

$$M(\psi(\cdot, t)) := \int_{\mathbb{R}^d} |\psi(\mathbf{x}, t)|^2 d\mathbf{x} \equiv M(\psi(\cdot, 0)) = 1, \quad t \geq 0, \tag{1.4}$$

and

$$E(\psi(\cdot, t)) := \int_{\mathbb{R}^d} \left[\frac{1}{2}|\nabla\psi(\mathbf{x}, t)|^2 + V(\mathbf{x})|\psi(\mathbf{x}, t)|^2 + \frac{\beta}{2}|\psi(\mathbf{x}, t)|^4\right] d\mathbf{x} \equiv E(\psi(\cdot, 0)), \quad t \geq 0. \tag{1.5}$$

The stationary solutions, which are a particular class of interesting solutions for (1.1), are given by the ansatz $\psi(\mathbf{x}, t) = \phi(\mathbf{x})e^{-i\mu t}$, where $(\mu, \phi)$ satisfies the following nonlinear eigenvalue problem:

$$\mu\phi(\mathbf{x}) = -\frac{1}{2}\Delta\phi(\mathbf{x}) + V(\mathbf{x})\phi(\mathbf{x}) + \beta|\phi(\mathbf{x})|^2\phi(\mathbf{x}), \quad \mathbf{x} \in \mathbb{R}^d, \tag{1.6}$$

under a unitary mass normalization condition

$$||\phi||_2^2 := \int_{\mathbb{R}^d} |\phi(\mathbf{x})|^2 \, d\mathbf{x} = 1. \tag{1.7}$$

Here $\mu$ is known as the chemical potential of the condensate given as

$$\mu = \mu(\phi) = \int_{\mathbb{R}^d} \left[\frac{1}{2}|\nabla\phi(\mathbf{x})|^2 + V(\mathbf{x})|\phi(\mathbf{x})|^2 + \beta|\phi(\mathbf{x})|^4\right] d\mathbf{x} = E(\phi) + \int_{\mathbb{R}^d} \frac{\beta}{2}|\phi(\mathbf{x})|^4 d\mathbf{x}.$$

Among all the eigenstates, the most important one is the so-called *ground state (GS)* [9,11,43], which minimizes the energy functional under the unitary mass constraint:

$$\phi_g := \arg\min_{\|\phi\|_2=1} E(\phi), \qquad E_g := E(\phi_g). \tag{1.8}$$

We remark here that (1.6) is the Euler-Lagrange equation for minimizing the energy (1.5) under the constraint of mass conservation (1.7), where the chemical potential $\mu$ is viewed as a Lagrange multiplier for the mass conservation. In this paper, we mainly focus on how to efficiently solve the GS solution by a deep learning method.

To numerically compute the GS of BEC, there are already quite many powerful methods. The most traditional method is the gradient flow with discrete normalization method or known as the imaginary-time evolution method [11,18,39], and later on many more methods like the constrained optimization techniques [4,20,21] and the nonlinear eigenvalue solvers [1,16] have also been developed. These methods though mathematically elegant, are all based on traditional spatial discretizations of the wave function in the computational domain. Therefore, they are basically limited to rather low-dimensional setup. For the original many-body Schrödinger equation or high-dimensional GPE, the curse-of-dimensionality will preclude their availability for applications.

In recent years, the deep neural network (DNN) has made enormous success in a wide scientific computing area. It is hopefully to overcome the curse-of-dimensionality, and this gives it great potential to be considered for solving the quantum many-body system. Although the final and meaningful target might be the many-body Schrödinger model, we need to begin with the low-dimensional problem to develop the detailed method and understand its performance. Thus in this work, we would like to begin with the development of the DNN for computing GS of the GPE. This kind of trend in fact has already begun among physicists recently. For instance, the work [46] first considers the parameter generation of GS by using deep convolution neural networks for 2-dimensional (2D) GPE and 1-dimensional (1D) two-component case. The Gaussian process is used in [6] later to reduce the required data for training by incorporating Bayesian optimization. The work [5] investigates the theory-guided neural network for 1D GPE with spin-orbit coupling. These existing DNN approaches are to some extent based on the supervised learning which requires data

of the unknown for label, and the data is obtained from the traditional numerical methods like the imaginary-time evolution. Such requirement of data seems not possible or very expensive again for high-dimensional or multi-component problems.

In this work, we propose to compute the GS via an unsupervised learning approach. We shall propose a suitable DNN which we call *normalized DNN (norm-DNN)*, to approximate the GS and directly solve the optimization problem (1.8) without calling data from traditional numerical methods. In fact with norm-DNN, we can convert the original constraint optimization in functional space into an unconstrained minimization in finite dimensional parameter space, where the unitary mass normalization (1.7) is naturally provided by the network by introducing a normalization layer. By utilizing some mathematical features of the solution, some other key techniques including a shift layer in the network, Gaussian initialization and a regularization in the loss are introduced to provide efficient training process and accurate approximations of GS. Investigations through extensive numerical experiments will firstly be made in a systematical way for 1D GPE and then extended to the high-dimensional case and the two-component case. The numerical results will also show that the application of standard neural network to (1.8) could fail to produce GS, which illustrates the importance of norm-DNN. The proposed norm-DNN is also extended to compute the first excited states of BEC. Furthermore, parameter generations of the proposed norm-DNN will be given in the end and comparisons will be made with traditional supervised learning DNN.

The rest of the paper is organized as follows. In Section 2, we shall review some useful features of the GS and define the structure of norm-DNN. Section 3 investigates the norm-DNN for 1D GS problem, and Section 4 considers extensions to solve GS of BEC in high-dimensions and two-component case and to compute the first excited states. We shall present the parameter generation and comparisons with existing machine learning methods in Section 5. Some conclusions are summarized in Section 6.

## 2. Normalized deep neural network for ground state

In this section, we shall first go through some preliminary for the ground state (GS) which will be helpful for computations later. Then, we shall propose our main neural network and present the computational setup for the training.

### 2.1. Feature of GS and traditional numerical approach

Under (1.2) and the case $\beta \geq 0$, GS always exists for (1.8). Moreover, we have the following key mathematical features for it [9,41,43]:

(i) GS is unique up to a phase shift, i.e., $e^{i\xi}\phi_g$ for any $\xi \in \mathbb{R}$ is also a GS; so $\phi_g$ can be chosen nonnegative;
(ii) If the potential $V$ is smooth, then $\phi_g \in C^\infty$ and it is exponentially decaying at far field;
(iii) If $V$ has radial symmetry, then $\phi_g$ is also radially symmetric (so it is an even function in 1D).

The design of our neural network and computations later will benefit from these features.

Before we introduce the network, we first review the traditional numerical method. The most traditional and popular way to compute GS is the gradient flow with discrete normalization (GFDN) method or known as the imaginary-time evolution method: set a time series $0 = t_0 < t_1 < t_2 < \cdots < t_n < \cdots$ with an initial guess $\phi(\mathbf{x}, 0) = \phi_0(\mathbf{x})$ and evaluate

$$\phi_t = -\frac{1}{2}\frac{\delta E(\phi)}{\delta \phi} = \frac{1}{2}\nabla^2\phi - V\phi - \beta|\phi|^2\phi, \quad t_n < t < t_{n+1}, \quad \phi(\mathbf{x}, t_{n+1}) = \frac{\phi(\mathbf{x}, t_{n+1}^-)}{||\phi(\cdot, t_{n+1}^-)||_2}, \quad n \geq 0, \tag{2.1}$$

till a steady state is reached. To numerically implement (2.1), one can truncate the whole space to a bounded domain and impose zero boundary conditions. Then a basic scheme for discretizating the partial differential equation (PDE) in (2.1) is to apply the backward Euler finite difference in time and the sine pseudospectral method in space. We refer the readers to [9,11] for the detailed scheme, and we shall use this scheme as the benchmark for reference GS solutions in our numerical experiments later. For more advanced traditional schemes, we refer to [1,4,16,20,21] and references therein.

The traditional approaches contain two drawbacks. (i) The first drawback is of course the curse-of-dimensionality for discretizing the PDE in high dimensions. The storage of mesh grids and computational costs can be expensive. (ii) The second drawback comes from the fact that the GS solution $\phi_g$ depends on the free parameters in the model, e.g., the value of $\beta$ and the values of the parameters in the potential (1.3), and for each single group of chosen parameters one will have to run the GFDN (2.1) for the corresponding $\phi_g$. This is also quite costly and tedious in applications.

These drawbacks motivate us to study the deep neural network approach for the GS problem. We shall propose a normalized deep neural network (norm-DNN) for the effective approximation of GS, and the norm-DNN can be trained to directly provide any GS within a prescribed parameter plane.

### 2.2. Normalized deep neural network

The concept of neural network exists for a long time. By including multiple hidden layers to make it deep, the deep neural network (DNN) mimics the function of a human brain. It can be trained to learn complex relations with substantial amounts of data, which has found great success in computer vision and natural language processing [14,37,38,52]. Here let us first provide a brief introduction to the network architecture of DNN. In general, given an input $\mathbf{y}_0 \in \mathbb{R}^{n_0}$ with $n_0 \in \mathbb{N}^+$, a feedforward DNN with $L \in \mathbb{N}$ hidden layers reads as a composite function of the form:

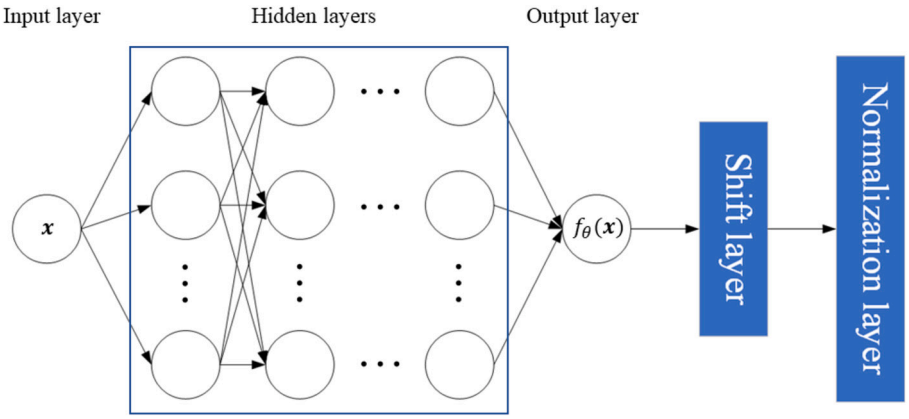


**Fig. 1.** Illustration of the architecture of norm-DNN.

$$f_\theta(\mathbf{y}_0) = \mathbf{F}_{L+1} \circ \sigma \circ \mathbf{F}_L \circ \sigma \circ \cdots \circ \mathbf{F}_2 \circ \sigma \circ \mathbf{F}_1(\mathbf{y}_0), \quad \theta := \{W_l, b_l\}, \tag{2.2}$$

where all the $\mathbf{F}_l : \mathbb{R}^{n_{l-1}} \to \mathbb{R}^{n_l}$ are some affine transforms:

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

with $W_l \in \mathbb{R}^{n_l \times n_{l-1}}$, $b_l \in \mathbb{R}^{n_l}$ for $n_l \in \mathbb{N}^+$. The last one $\mathbf{F}_{L+1}$ is called as the output layer and $\mathbf{F}_l$ for $1 \le l \le L$ are called as the hidden layers, with $n_l$ interpreted as the number of neurons in the $l$-th layer. In particular, $n_0$ and $n_{L+1}$ denote the dimensions of the input and output, respectively. Moreover, $\sigma : \mathbb{R} \to \mathbb{R}$ is known as the activation function that acts on each component of $\mathbf{F}_l$, i.e., $\sigma \circ \mathbf{F}_l \in \mathbb{R}^{n_l}$. For simplicity, we shall restrict to consider the fully connected neural network (FCNN) and assume that all the hidden layers use the same number of neurons, i.e., $W = n_1 = \ldots = n_L$. We refer the total number of hidden layers $L$ as 'depth', and we refer the number of neurons $W$ in the hidden layers as 'width'.

The power of DNN for approximating a general nonlinear function has received many theoretical supports. The early universal approximation theorem [19] already states that for any continuous function $h : [0,1]^d \to \mathbb{R}$ and any $\varepsilon > 0$, there exists a neural network $f_\theta(\mathbf{x})$ with at least one hidden layer and a sigmoidal activation function such that

$$\|h(\mathbf{x}) - f_\theta(\mathbf{x})\|_{L^\infty} < \varepsilon. \tag{2.3}$$

Later, the theorem has been extended [32–34] to cover cases of other types of target functions and activation functions. For smooth target functions, more delicate error bounds have been established to reveal how the error depends on the depth and width of DNN [13,35,51,54]. For further recent analytical achievements, we refer to [25,31,44,55]. The mathematical features of the GS from Section 2.1 show that $\phi_g$ certainly belongs to the case that can be covered by the theoretical results, so an effective DNN approximation for our GS problem does exist, and then the primary focus of this paper is to address the practical way for computing it.

On the other hand, DNN in recent years has been extensively applied and developed to solve PDEs [23,24,26,29,49,53,58]. To tackle the GS problem (1.8), one could think of using these PDE approaches for instance, the physics-informed neural networks (PINNs) [49] to solve the GFDN (2.1). However, we find this may not be a good option. To apply methods like PINNs to (2.1), besides the special efforts needed to treat the piece-wisely defined PDE and the normalization in (2.1), one will need to fix the time interval $t \in [0, T]$ to solve the PDE for the solution. Unfortunately, the terminal time $T > 0$ for the GFDN (2.1) to reach a steady state is not known in advance. Improperly prescribed $T$ may either lead to an inaccurate final state or a waste of computational resource. Thus, in this work we propose to consider DNNs directly for the minimization (1.8), and practical ways will be presented to find the one that theoretically exists in (2.3).

To construct an effective DNN to approximate the GS $\phi_g$, the basic DNN that we introduce in this paper is the following **normalized DNN (norm-DNN)**:

$$\phi_\theta(\mathbf{x}) = \mathcal{N} \circ \mathcal{T} \circ \mathbf{F}_{L+1} \circ \sigma \circ \mathbf{F}_L \circ \sigma \circ \cdots \circ \mathbf{F}_2 \circ \sigma \circ \mathbf{F}_1(\mathbf{x}), \quad \mathbf{x} \in \mathbb{R}^d, \tag{2.4}$$

where $\mathcal{N}(\mathbf{y}) = \mathbf{y}/||\mathbf{y}||_2$ with $||\mathbf{y}||_2 = \sqrt{\int_\Omega |\mathbf{y}|^2 d\mathbf{x}}$ and $\mathbf{y} = \mathbf{y}(\mathbf{x}) : \mathbb{R}^d \to \mathbb{R}^{n_{L+1}}$ is the *normalization layer* to ensure that the output of norm-DNN always meets the mass constraint in (1.8). Moreover in (2.4), we have introduced another special layer that we call the *shift layer*: $\mathcal{T}(\mathbf{y}) = \mathbf{y} - \min_{\mathbf{x}}(\mathbf{y})$, $\mathbf{y} = \mathbf{y}(\mathbf{x}) : \mathbb{R}^d \to \mathbb{R}^{n_{L+1}}$. Here the minimization is acting on each component of $\mathbf{y}$. It provides a non-negativity restriction and helps the norm-DNN (2.4) in a minimization process to turn to the GS instead of an excited state. For the scalar/single-component BEC case (1.8), we would clearly take $n_{L+1} = 1$. Fig. 1 illustrates the basic network architecture of norm-DNN. For the activation function, a classical choice in machine learning would be ReLU, but DNN with ReLU is non-differentiable. Concerning the fact that $\phi_g$ is smooth and the energy (1.8) contains a gradient, we choose to consider smooth activation functions such as $\sigma = \tanh$. The detailed strategy to choose $\sigma$ can be problem-dependent, and a better choice for some special occasion like the optical lattice potential case can significantly increase the efficiency of norm-DNN. This will be clarified later.

We illustrate here that the norm-DNN (2.4) with the proposed two additional layers will not break the automatic differentiation feature of DNN. The norm-DNN (2.4) can be denoted as $\phi_\theta(\mathbf{x}) = \mathcal{N}(\mathbf{y}(\mathbf{x};\theta))$ with $\mathbf{y}(\mathbf{x};\theta) = \mathcal{T} \circ f_\theta(\mathbf{x}) = \mathbf{f}_\theta(\mathbf{x}) - \min_{\mathbf{x}} \mathbf{f}_\theta(\mathbf{x})$ and $f_\theta(\mathbf{x})$ the standard DNN (2.2). It is well-known that the derivatives of $f_\theta$ such as $\partial_{\mathbf{x}} f_\theta$ and $\partial_\theta f_\theta$ can be computed by the automatic differentiation in Python for DNN. Now we consider the computation of $\partial_{\mathbf{x}} \phi_\theta$. Since $\|\mathbf{y}\|_2$ and $\min_{\mathbf{x}} f_\theta(\mathbf{x})$ are independent of $\mathbf{x}$, we can find

$$\partial_{\mathbf{x}} \phi_\theta = \frac{1}{\|\mathbf{y}\|_2} \partial_{\mathbf{x}} f_\theta.$$

For $\partial_\theta \phi_\theta$, we have

$$\partial_\theta \phi_\theta = \frac{\partial_\theta \mathbf{y}}{\|\mathbf{y}\|_2} - \frac{\mathbf{y} \int_\Omega \partial_\theta \mathbf{y} \cdot \mathbf{y} d\mathbf{x}}{\|\mathbf{y}\|_2^3},$$

with $\partial_\theta \mathbf{y} = \partial_\theta f_\theta(\mathbf{x}) - \min_{\mathbf{x}} \partial_\theta \mathbf{f}_\theta(\mathbf{x})$. Then, $\partial_{\mathbf{x}} \phi_\theta$ and $\partial_\theta \phi_\theta$ can be found out with $\partial_{\mathbf{x}} f_\theta$ and $\partial_\theta f_\theta$ available. Thus, the derivatives of the norm-DNN (2.4) can still be computed via the automatic differentiation.

To apply the deep learning algorithm, we need to define the optimization objective, also known as the loss function for norm-DNN. For the GS problem (1.8), it is now natural to consider the energy functional $E$ as the loss, i.e., Loss $:= E(\phi_\theta)$, and then $\phi_\theta \approx \phi_g$ would be the target norm-DNN (2.4) that we look for. For a practical implementation, the whole space integration in Loss/energy (1.5) has to be truncated to a finite domain $\Omega \subset \mathbb{R}^d$ and the integral has to be discretized. Therefore, the practical loss that we will use is defined as

$$\text{Loss} = \text{Loss}(\theta) := \frac{|\Omega|}{N} \sum_{j=1}^{N} \left[ \frac{1}{2} |\nabla \phi_\theta(\mathbf{x}_j)|^2 + V(\mathbf{x}_j) |\phi_\theta(\mathbf{x}_j)|^2 + \frac{\beta}{2} |\phi_\theta(\mathbf{x}_j)|^4 \right], \tag{2.5}$$

where $|\Omega|$ denotes the measure of the domain and $\{\mathbf{x}_j\}_{j=1}^N \subset \Omega$ are the quadrature points for approximating the integral with $N > 0$ the total number of points. Since $\phi_g$ is exponentially localized, as long as $\Omega$ is large enough the truncation error of the domain would be negligible. Note that no boundary constraint is imposed in the loss. The quadrature points here are interpreted as the training set for DNN. For a high-dimensional problem, they are often generated by a Monte Carlo method to overcome the curse-of-dimensionality [26]. The price is of course the low accuracy from the random sampling and the result might be polluted by noise. In this work, let us first consider $\{\mathbf{x}_j\}_{j=1}^N$ as fixed equally distributed grid points in $\Omega$, then (2.5) is the Riemann sum and so the quadrature error of LOSS can be very small. This will allow us to focus on studying the performance of the proposed norm-DNN on GS computing without the interference from the sampling noise. For the high-dimensional examples, we will test the Monte Carlo sampling. More sampling issues would be investigated and improved in future by incorporating recent developments like the adaptive sampling [59] or the quasi-Monte Carlo sampling [42].

**Remark 2.1.** Here we briefly discuss the computational complexity of norm-DNN (2.4) compared with the traditional method like GFDN (2.1). The implementation of GFDN needs a mesh grid, which means with $N$ grid points in each direction, the total number of unknowns is then $N^d$. Therefore, its storage and computational costs grow exponentially with respect to the dimensionality $d$. This is not the case for norm-DNN (2.4), whose unknowns are the matrices and vectors in the layers. Note that only the size of $W_1$ and $b_1$ in the first layer $F_1$ of (2.4) depends linearly on $d$, while the others can be totally independent of $d$. Additionally, the total error of traditional methods like GFDN is an accumulation of discretization error in each dimension, so usually the error constant increases along with $d$. For the DNN approach, such approximation error is believed to be independent or weakly dependent on $d$ [54].

Now with (2.4) and (2.5), we can turn the original GS problem (1.8) which is a constraint optimization in functional space, into an unconstrained minimization in finite dimensional parameter space, i.e.,

$$\min_\theta \{\text{Loss}(\theta)\}. \tag{2.6}$$

To get the optimized norm-DNN (2.4) for (2.6), the basic approach 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 $\tau$ denotes the learning rate and $\theta^0$ is an initial guess. For the high-dimensional case of (2.5) where the number of training points may be very large, the stochastic gradient descent (SGD) method or the mini-batch gradient descent (m-BGD) method could also be applied to release the computational costs per iteration. However, SGD and m-BGD may have extra difficulties to provide fast and reliable convergence for training [22,50]. To avoid the additional troubles so that we can focus on the development of the network in this paper, the Adam optimizer [36] will be utilized for (2.6), which is a special optimization method widely considered in deep learning. Moreover, we will employ another commonly used training strategy in machine learning: gradually reducing the learning rate during the training process [30,56]. This can make the training process more stable and accelerate convergence. More precisely, here the initial learning rate $\tau$ is set as $10^{-3}$ and it decays after every 100 steps with a base of 0.99. We remark that the very recent work [12] could be an alternative DNN way for the GS problem (1.8).

To initialize the norm-DNN parameters for optimization (2.6), i.e., $\theta^0$, we employ the Xavier method [28] which generates the initialization in a random way. The training stops when the mean value of the loss function in adjacent 100 iterations becomes relatively stable, i.e., with a given threshold $tol > 0$,

**Table 1**
Summary of notations and parameters for the network.

| Notation | Meaning |
|---|---|
| $L$ | Depth of the network |
| $W$ | Width of the network |
| $\tau$ | Initial value of learning rate |
| $tol$ | Threshold to stop training |
| $N_x$ | Number of training points in $x$-direction |
| $N_y$ | Number of training points in $y$-direction |

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

To quantify the accuracy, we shall compute the relative $L^2$ error

$$\text{error} := \left\| \phi_\theta - \phi_g \right\|_2 / \left\| \phi_g \right\|_2 \tag{2.7}$$

between the exact solution and the approximate solution from norm-DNN. The $L^2$-norm in (2.7) will be calculated as the square-root of the Riemann sum over a finer grid (much finer than the grid for training in (2.5)) which can be interpreted as the test set. For numerical studies throughout the paper, the number of test points is 2048 for 1D problems and is $400 \times 400$ for 2D problems. The exact solution $\phi_g$ will be obtained from GFDN (2.1), and owning to the fact (i) in Section 2.1, it will be determined up to a sign. Notations for the parameters are summarized in Table 1, and the value of them will be specified in each experiment later.

All the numerical experiments in this paper are programmed in PyTorch sequentially and conducted on a laptop with an Intel Core i9-12950HX processor and 32 GB of memory. The codes are available in https://github.com/1761121438/Norm-DNN-for-computing-the-ground-state-of-BEC.git.

## 3. Application to 1-dimensional problems

Though DNN is meant for high-dimensional problems which we would eventually go for, here we would like to begin with the 1D case to clarify the basics and investigate the performance of norm-DNN. The GS problem by norm-DNN (2.6) in 1D now reads: $\mathbf{x} = x \in \mathbb{R}$ and

$$\theta^* = \arg\min_\theta \frac{|\Omega|}{N_x} \sum_{j=1}^{N_x} \left[ \frac{1}{2} |\nabla \phi_\theta(x_j)|^2 + V(x_j) |\phi_\theta(x_j)|^2 + \frac{1}{2} \beta |\phi_\theta(x_j)|^4 \right]. \tag{3.1}$$

We shall fix $\Omega = (-12, 12)$, $\beta = 400$ in this section. With numerical experiments, we shall illustrate in a sequel the superiority and necessity of the normalization layer $\mathcal{N}$ and the shift layer $\mathcal{T}$ in norm-DNN, the acceleration of convergence with pre-training and the influence of the network architecture. Then, we will present some special strategies for the optical lattice potential case.

### 3.1. Normalization layer

In norm-DNN (2.4), the normalization layer $\mathcal{N}$ is proposed to provide the unitary mass in (1.8). To show its superiority, here we consider the usual soft-constraint way as a comparison. That is to consider the standard DNN (2.2) to approximate GS and impose the mass constraint as a penalty term in the loss function (2.5). The corresponding loss function for the comparison under standard DNN then reads

$$\mathrm{Loss}(\theta) := E(f_\theta) + \eta \left( \|f_\theta\|_2 - 1 \right)^2,$$

where $f_\theta$ represents the output of (2.2). Let $V(x) = \frac{1}{2}x^2$, $\sigma = \tanh$, and the hyper-parameters are set as $L = 4$, $W = 70$, $tol = 10^{-6}$, $N_x = 128$, $\eta = 100$, which are not specifically tuned either for DNN or for norm-DNN. The relative error along with iterations, the exact and numerical solutions for norm-DNN and DNN are shown in Fig. 2.

As it can be seen from Fig. 2, the standard DNN (2.2) with penalty is ineffective under the set of parameters. It may work by carefully tuning the hyper-parameters especially the weight $\eta$ for penalty, but this could be relatively involved and tedious. On the contrary, the norm-DNN can accurately and efficiently capture the GS. This example illustrates here that the proposed hard constraint $\mathcal{N}$ goes beyond the typical soft penalty-type constraint, which makes norm-DNN more effective for learning GS.

### 3.2. Shift layer

From the review in Section 2.1, it is known that GS is determined up to a phase shift. Since our computational parameters in norm-DNN are all real-valued, a valid $\phi_\theta$ resulting from the training (3.1) should be an approximation of either $|\phi_g|$ or $-|\phi_g|$. However, a gradient flow type optimization method in general cannot guarantee to get the global minimizer/GS, because the GS problem (1.8) is a well-known non-convex optimization [9] and the minimization for norm-DNN in general is far away from being convex as well. In addition, the Xavier initialization [28] and the SGD (if applied) contain randomness, so it could happen that part of $\phi_\theta$ converges

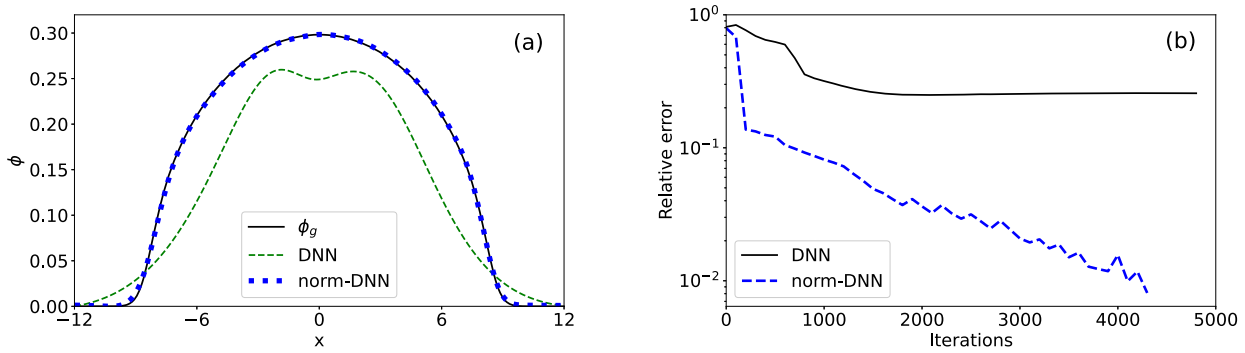


**Fig. 2.** Results for GS problem (1.8) with norm-DNN (2.4) and DNN (2.2): (a) exact and numerical solutions ($\|f_\theta\|_2 = 0.7928$ for DNN and $\|\phi_\theta\|_2 = 1$ for norm-DNN); (b) error during the iterations.

**Table 2**
Error (2.7) for 1D GS problem with or without shift layer $\mathcal{T}$.

| Error (2.7) | $L$ \ $W$ | 10 | 50 | 70 |
|---|---|---|---|---|
| with $\mathcal{T}$ | 1 | 5.70E-2 | 5.90E-2 | 6.01E-2 |
| | 2 | 6.34E-2 | 4.90E-2 | 1.51E-2 |
| | 3 | 4.38E-2 | 6.07E-3 | 1.72E-2 |
| | 4 | 4.63E-2 | 7.10E-3 | 1.05E-2 |
| without $\mathcal{T}$ | 1 | 1.43E+0 | 1.42E+0 | 1.40E+0 |
| | 2 | 1.40E+0 | 1.43E+0 | 1.40E+0 |
| | 3 | 1.43E+0 | 1.40E+0 | 1.40E+0 |
| | 4 | 1.43E+0 | 1.40E+0 | 1.33E+0 |

**Table 3**
Numerical energy $E(\phi_\theta)$ for 1D GS problem with or without shift layer $\mathcal{T}$ (exact $E(\phi_g) = 21.3601$).

| $E(\phi_\theta)$ | $L$ \ $W$ | 10 | 50 | 70 |
|---|---|---|---|---|
| with $\mathcal{T}$ | 1 | 21.3874 | 21.3922 | 21.3903 |
| | 2 | 21.4010 | 21.3755 | 21.3522 |
| | 3 | 21.3715 | 21.3513 | 21.3530 |
| | 4 | 21.3734 | 21.3556 | 21.3517 |
| without $\mathcal{T}$ | 1 | 22.1470 | 22.2626 | 22.1150 |
| | 2 | 23.0825 | 23.2284 | 23.1349 |
| | 3 | 23.0455 | 23.5141 | 23.6430 |
| | 4 | 22.9423 | 23.3252 | 23.4760 |

to $|\phi_g|$ and some other part converges to $-|\phi_g|$. This phenomenon will be observed in the coming numerical experiment and the obtained $\phi_\theta$ will only be a local minimum/excited state instead of the correct GS. To overcome this issue, here comes the shift layer $\mathcal{T}$ that we insert between the output layer and the normalization layer $\mathcal{N}$ in the norm-DNN (2.4). For the 1D GS problem, it is now defined as

$$\mathcal{T}(y) = y - \min_x(y), \quad y = y(x) : \mathbb{R} \to \mathbb{R}, \tag{3.2}$$

in order to provide a non-negativity restriction on the norm-DNN:

$$\phi_\theta(x) = \mathcal{N} \circ \mathcal{T} \circ \mathbf{F}_{L+1} \circ \sigma \circ \mathbf{F}_L \circ \sigma \circ \cdots \circ \mathbf{F}_2 \circ \sigma \circ \mathbf{F}_1(x). \tag{3.3}$$

Then the norm-DNN (3.3) will always go to a non-negative function through the training (3.1). Now let us first investigate its performance for computing GS under the harmonic oscillator potential case, and we fix the activation function as $\sigma = \tanh$.

We take $V(x) = \frac{1}{2}x^2$ for the numerical experiment. Setting the hyper-parameters as $L \in \{1,2,3,4\}$, $W \in \{10,50,70\}$, $tol = 10^{-6}$, $N_x = 128$, we solve the 1D GS problem as described before. To show the necessity of the shift layer $\mathcal{T}$, we compare the performance of the norm-DNN with $\mathcal{T}$ as defined in (3.3) or without $\mathcal{T}$ (or set $\mathcal{T} = id$). The relative errors (2.7) under the two cases are presented in Table 2. The numerical energy $E(\phi_\theta)$ for norm-DNN with or without $\mathcal{T}$ is shown in Table 3. Here the exact GS energy $E(\phi_g)$ is 21.3601, which can be obtained from GFDN.

It is clear from the results in Table 2 that in the absence of $\mathcal{T}$, the relative errors always exceed one and so the approximations are completely invalid. In contrast, when $\mathcal{T}$ (3.2) is included, the relative errors of norm-DNN (3.3) become much smaller, reaching as low as $6 \times 10^{-3}$. From Table 3, we find that $E(\phi_\theta)$ of norm-DNN with $\mathcal{T}$ is closer to $E(\phi_g)$, while $E(\phi_\theta)$ of norm-DNN without $\mathcal{T}$

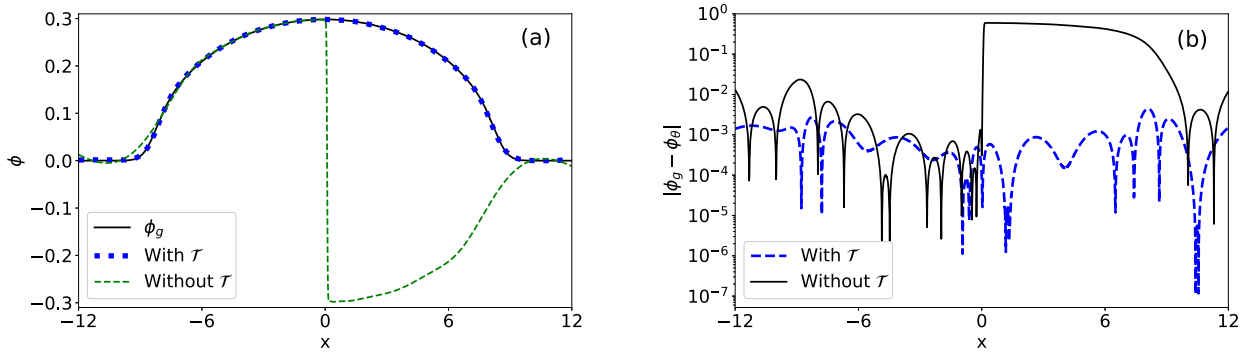


**Fig. 3.** Results for GS problem under $L = 3$, $W = 50$: (a) exact solution and numerical solution with or without $\mathcal{T}$; (b) pointwise error of numerical solution.

**Table 4**
Number of iterations, computational time and relative error (2.7) for norm-DNN (3.3) with or without Gaussian pre-training.

| | Iterations | Time | Error (2.7) |
|---|---|---|---|
| $\phi_\theta$ | 12100 | 45 s | 6.04E-2 |
| $\phi_\theta^G$ | 6800 | 29 s | 3.11E-2 |

is larger. This shows that norm-DNN without $\mathcal{T}$ will produce some excited states. Under $L = 3$, $W = 50$, we show the exact solution $\phi_g$ and the numerical solution $\phi_\theta$ with or without using $\mathcal{T}$ in Fig. 3. It can be observed that without $\mathcal{T}$ to fix the sign, the norm-DNN $\phi_\theta$ will partially go to the positive GS and partially go to the negative one, and in the end the training will produce an excited state. This highlights the crucial role of the shift layer $\mathcal{T}$ in enabling norm-DNN to accurately capture the GS. We remark that though we have used the Adam method for optimization here, using SGD will give the same results and conclusion.

**Remark 3.1.** Instead of imposing the shift layer $\mathcal{T}$ in the norm-DNN (3.3), an alternative approach is to make use of the symmetry feature of GS. We can add a regularization term in the loss to restrict $\phi_\theta(x)$ being even in space, and then we solve the following minimization problem:

$$\min_\theta \{\text{Loss}(\theta) + \text{Even}(\theta)\}, \quad \text{where} \quad \text{Even}(\theta) = \frac{\delta}{N_e}\sum_{j=1}^{N_e} |\phi_\theta(x_j^e) - \phi_\theta(-x_j^e)|^2, \tag{3.4}$$

with a weight $\delta > 0$ and some training points $\{x_j^e\}_{j=1}^{N_e} \subset \Omega \cap \mathbb{R}^+$. Then, (3.3) without $\mathcal{T}$ will also lead to the GS. The detailed results are omitted here for brevity. Note in high dimension when the radial symmetry is absent, the shift layer then becomes the only option.

### 3.3. Gaussian pre-training

In the linear regime of BEC, i.e., $\beta = 0$ in (1.1), the GS under a harmonic potential can be found out explicitly as a Gaussian function [9]. Therefore, traditional methods like GFDN commonly use Gaussian functions as the initial data in (2.1) so that the flow can avoid local minimums and can reach the steady state faster. Here we borrow this idea and implement it into our norm-DNN.

What we propose is to utilize a Gaussian function to pre-train the norm-DNN (3.3) to get the initialization $\theta^0$ for the Adam method to solve (2.6). More precisely, with the same numerical example $V(x) = \frac{1}{2}x^2$ and $\beta = 400$ as before, we train a norm-DNN (3.3) generated from Xavier initialization [28] for 1000 iterations to fit a normalized Gaussian $\phi_0 := e^{-\frac{x^2}{10}}/(5\pi)^{1/4}$. Then, we test the performance of the norm-DNN with $L = 2$, $W = 50$, and the other hyper-parameters are set as $tol = 10^{-5}$, $N_x = 128$. To illustrate the improvement in efficiency, we consider two cases for comparison: $\phi_\theta^G$ from (3.3) with the described Gaussian pre-training; $\phi_\theta$ from (3.3) without using the Gaussian pre-training. The total number of iterations used in the optimization/training, the computational time (in seconds) and the error (2.7) are shown in Table 4. Here and after, the computational time is counted after the pre-training. The profiles of the initial norm-DNNs, the final norm-DNNs and the training process are shown in Fig. 4. In addition, we emphasize that the Gaussian pre-training cannot replace the role of the shift layer $\mathcal{T}$. For this purpose, Fig. 4(b) shows the numerical solution $\phi^G$ from (3.3) with the Gaussian pre-training but without imposing $\mathcal{T}$.

From the numerical results, we have the following observations. Firstly, by the red line in Fig. 4(a), we notice that the initial value generated by the Xavier method is far away from the GS. The initial function in fact does not satisfy any mathematical properties of the GS, and our norm-DNN approach can still accurately get to the GS within a certain number of iterations. This demonstrates the robustness of the proposed norm-DNN (3.3). Secondly, as shown in Table 4, the pre-training with a Gaussian allows norm-DNN to converge to a more accurate GS with much less iterations and computational time. Fig. 4(e,f) illustrate that during the entire training process, norm-DNN with pre-training converges much faster in terms of both iterations and time. The pointwise accuracy of

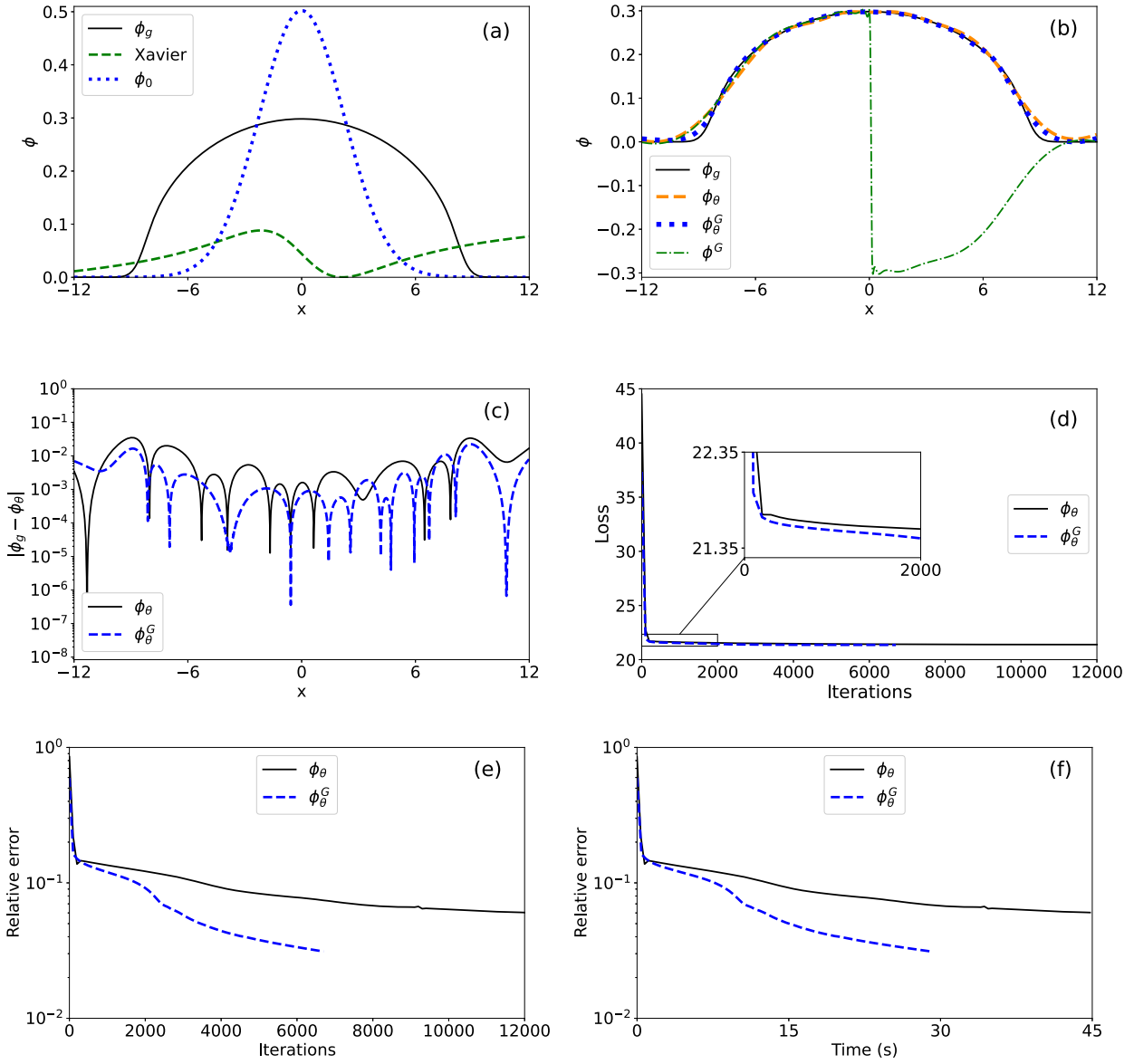


**Fig. 4.** GS problem: (a) initial profile of norm-DNN from Gaussian pre-training or Xavier method [28]; (b) exact $\phi_g$, norm-DNN $\phi_\theta^G$ with pre-training and $\mathcal{T}$, norm-DNN $\phi_\theta$ with $\mathcal{T}$ but without pre-training, norm-DNN $\phi^G$ with pre-training but without $\mathcal{T}$; (c) pointwise error; (d,e,f) loss and error (2.7) during the iterations or computational time. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

the pre-trained norm-DNN also becomes better as Fig. 4(c) indicates. Note in addition that the time needed by pre-training is very short, and this is done once for all. Thus, we conclude that the pre-training of norm-DNN can gain significantly more efficiency. As it is already available, we will apply directly the pre-trained norm-DNN in all the following experiments.

### *3.4. Influence of network architecture*

Although there is no clear theory to guarantee that more hidden layers and more neurons can bring DNNs closer to the global minimum, the convention in deep learning agrees that DNNs with larger architecture tend to gain more accuracy [17,26,49]. With the pre-trained initialization fixed, here we carry out a systematical numerical study on the influence of the network architecture for our norm-DNN (3.3). The hyper-parameters are set as $L \in \{1,2,3,4\}$, $W \in \{10,50,70\}$, $tol = 10^{-6}$, $N_x = 128$.

The errors (2.7) and the consumed training time of norm-DNN under different $W$ and $L$ are shown in Table 5. Clearly, we can see that both the errors and the training time are decreasing as the depth and width of the network increase. This general trend indicates that the ability of norm-DNN to approximate GS improves with the increase of the architecture.

**Table 5**
Error (2.7) and training time of norm-DNNs with different architecture.

| | $W = 10$ | $W = 50$ | $W = 70$ |
|---|---|---|---|
| $L = 1$ | 5.50E-2(109 s) | 5.86E-2(87 s) | 6.91E-2(170 s) |
| $L = 2$ | 5.27E-2(128 s) | 9.26E-3(84 s) | 6.79E-3(49 s) |
| $L = 3$ | 5.56E-2(116 s) | 7.54E-3(32 s) | 4.89E-3(28 s) |
| $L = 4$ | 4.23E-2(92 s) | 4.89E-3(26 s) | 4.57E-3(22 s) |

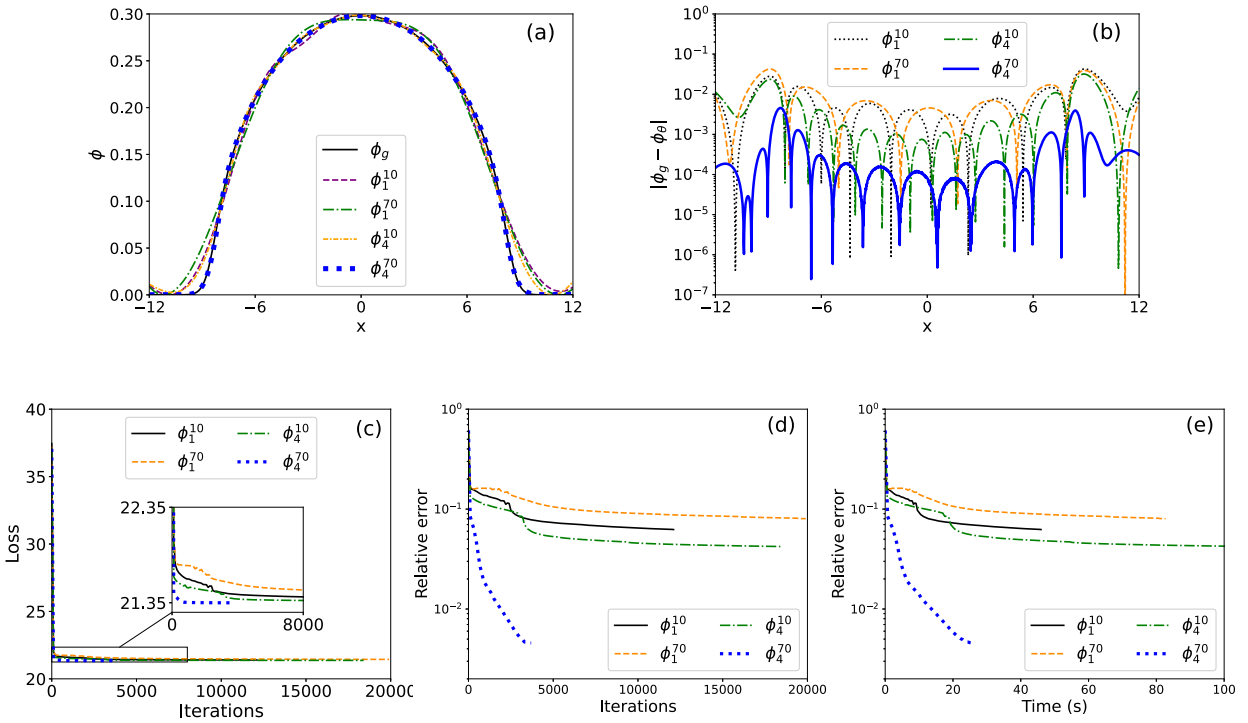


**Fig. 5.** Some norm-DNNs with different architecture for GS problem: (a) exact solution and numerical solution of norm-DNN; (b) pointwise error; (c,d,e) loss and error (2.7) during the iterations or computational time.

Denote $\phi_L^W$ for the norm-DNN (3.3) with width $W$ and depth $L$. The fitting effects of norm-DNN with $L \in \{1,4\}$, $W \in \{10,70\}$ for the GS are shown in Fig. 5(a) and their corresponding pointwise errors are shown in Fig. 5(b). It should be noted that if the width or depth is not sufficient, norm-DNN could remarkably deviate from the GS at the corners. Increasing the network architecture to $L = 4$ and $W = 70$ results in a very good pointwise approximation. One concern is that the increase of the architecture will lead to the increase of computational cost per iteration in the optimization for (3.1). However, as demonstrated in Fig. 5(c,d), when the network architecture is sufficiently large, norm-DNN in fact requires no more than 10000 iterations to reduce the relative error down to $4.57 \times 10^{-3}$. Consequently, the total training time is significantly reduced, though the time per iteration increases.

### 3.5. Optical lattice potential

The previous studies are for the harmonic potential case. Now, we consider the optical lattice potential (1.3) case which can contain high-frequency information through the additional trigonometric terms. This will result in some local oscillations in the corresponding GS, e.g., Fig. 6(a). This kind of high-frequency information in the target function may not be well captured by DNNs with 'tanh' as the activation function [17].

To investigate such issue in norm-DNN for the GS problem, we take $V(x) = \frac{1}{2}x^2 + 25\sin^2(\frac{\pi x}{4})$, $\beta = 400$ in (3.1) for the numerical experiment. The pre-training with the normalized Gaussian is applied as before, and the hyper-parameters are set as $L \in \{3,4\}$, $W = 10$, $tol = 10^{-6}$, $N_x = 128$. Then, the numerical results of the proposed norm-DNN (3.3) with $\sigma = \tanh$ and $L = 4$, $W = 10$ are shown in Fig. 6, where the results basically tell that the approximation is invalid. To improve the performance of norm-DNN in the optical lattice potential case, we propose to consider another activation function 'sin' in (3.3), i.e., $\sigma = \sin$, which brings naturally some oscillations to the network, and the corresponding numerical results are also shown in Fig. 6 as comparison. To further illustrate the gained accuracy and efficiency, we show the number of iterations, computational time and error (2.7) for the norm-DNN with $\sigma = \tanh$ or $\sigma = \sin$ in Table 6.

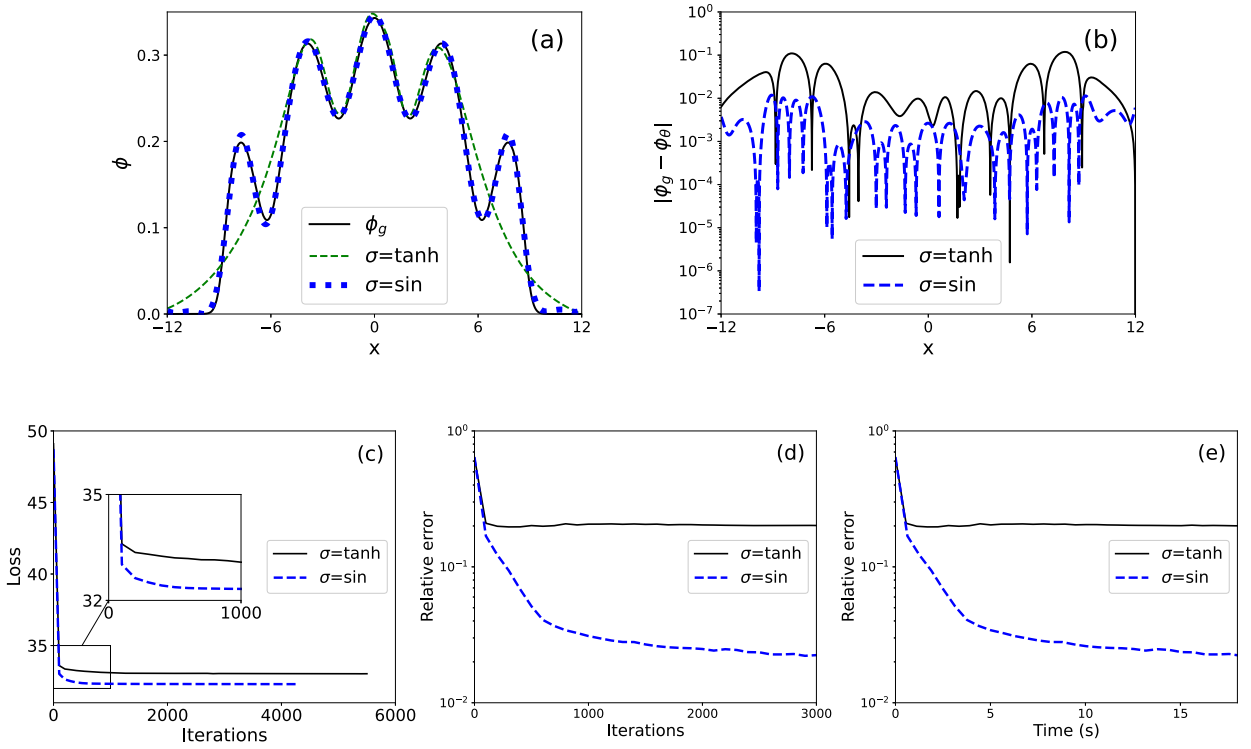


**Fig. 6.** norm-DNNs with different activation functions for GS problem: (a) exact solution and numerical solution of norm-DNN; (b) pointwise error; (c,d,e) loss and error (2.7) during the iterations or computational time.

**Table 6**
Number of iterations, computational time and error (2.7) of norm-DNN (3.3) with 'tanh' or 'sin' as the activation function.

| $\sigma$ | $W \times L$ | Iterations | Time | Error (2.7) |
|---|---|---|---|---|
| tanh | $10 \times 3$ | 27800 | 131 s | 1.97E-1 |
| | $10 \times 4$ | 5600 | 30 s | 1.98E-1 |
| sin | $10 \times 3$ | 5000 | 26 s | 2.23E-2 |
| | $10 \times 4$ | 4400 | 27 s | 1.97E-2 |

Clearly from the presented numerical results, we find that the norm-DNN with 'sin' is able to produce the correct approximation of GS in an efficient and accurate manner. In contrast, the norm-DNN with 'tanh' takes more iterations and more time for training, but it is still only able to fit three wave structures in the GS as shown in Fig. 6. The norm-DNN with 'sin' can accurately capture all the local oscillatory structures. Thus, we conclude that the 'sin' activation function is the suitable choice for norm-DNN to compute GS in the optical lattice potential case.

## 4. Application to extended problems

The norm-DNN method proposed for GS of 1-dimensional BEC in the previous section, along with the conclusions drawn, is similarly applicable to high-dimensional or multi-component models. With appropriate adjustments, it can also be applied to compute the first excited state. In this section, we are going to present these extensions. We shall first study the applications of norm-DNN to the two-dimensional and the three-dimensional GS problems. Then, we shall consider the application to the two-component GS problem and the computation of the first excited state in a sequel.

### *4.1. High-dimensional GS problem*

We now consider the GS problem (1.8) for the two-dimensional (2D) BEC, i.e., $d = 2$ and $\mathbf{x} = (x, y)^\top \in \mathbb{R}^2$ in (1.1). Based on our 1D study, the straightforward extension of the norm-DNN approach to 2D case then considers the optimization:

**Table 7**
The relative error (2.7) of norm-DNN for the 2D GS problem in Example 4.1 and Example 4.2.

| Example 4.1 | $W=10$ | $W=50$ | $W=70$ | Example 4.2 | $W=10$ | $W=50$ | $W=70$ |
|---|---|---|---|---|---|---|---|
| $L=1$ | 1.85E-1 | 1.51E-1 | 1.40E-1 | $L=1$ | 1.59E-1 | 1.63E-1 | 1.55E-1 |
| $L=2$ | 1.17E-1 | 4.40E-2 | 4.44E-2 | $L=2$ | 1.25E-1 | 6.41E-2 | 1.25E-1 |
| $L=3$ | 1.68E-1 | 2.36E-2 | 2.55E-2 | $L=3$ | 8.85E-2 | 3.14E-2 | 3.26E-2 |
| $L=4$ | 7.92E-2 | 2.85E-2 | 2.46E-2 | $L=4$ | 5.12E-2 | 2.86E-2 | 2.80E-2 |

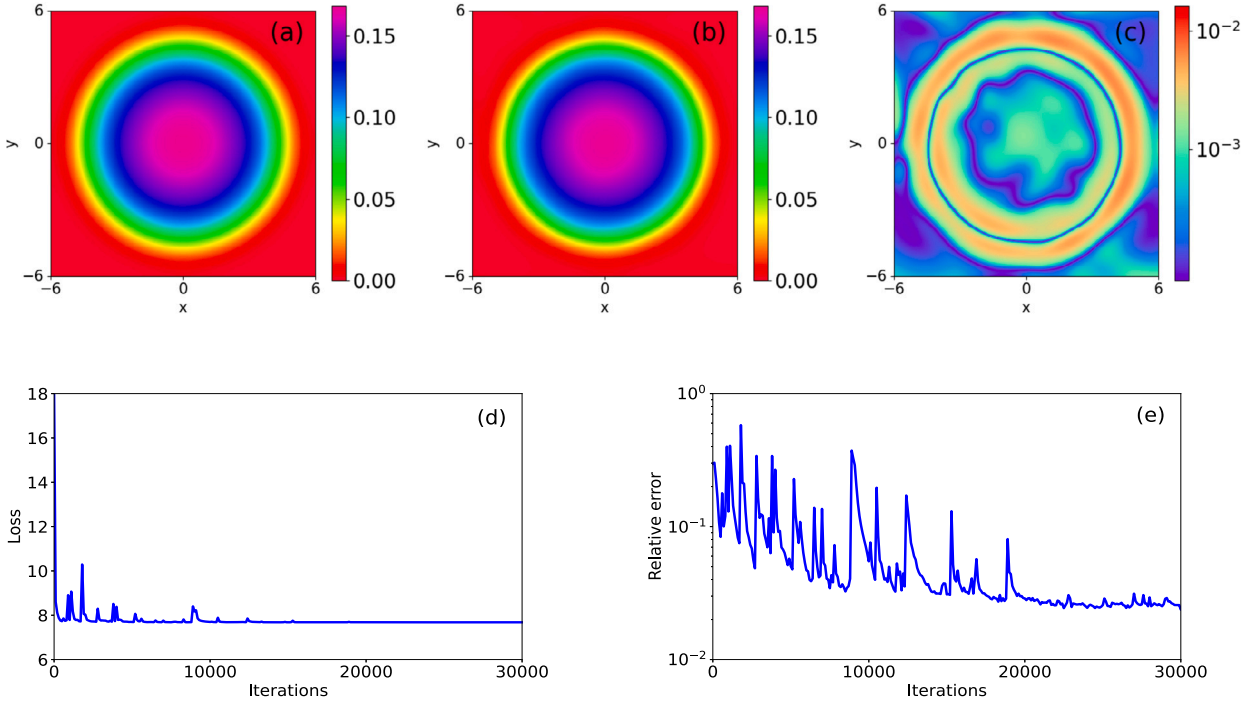


**Fig. 7.** Norm-DNN for 2D GS problem Example 4.1: exact GS (a); numerical solution of norm-DNN (b); pointwise error (c); loss (d) and error (2.7) (e) during the iterations.

$$\min_{\theta}\left\{\text{Loss}=\frac{|\Omega|}{N_x\times N_y}\sum_{j=1}^{N_x}\sum_{k=1}^{N_y}\left[\frac{1}{2}|\nabla\phi_\theta(x_j,y_k)|^2+V(x_j,y_k)|\phi_\theta(x_j,y_k)|^2+\frac{1}{2}\beta|\phi_\theta(x_j,y_k)|^4\right]\right\} \tag{4.1}$$

with

$$\phi_\theta(x,y)=\mathcal{N}\circ\mathcal{T}\circ\mathbf{F}_{L+1}\circ\sigma\circ\mathbf{F}_L\circ\sigma\circ\cdots\circ\mathbf{F}_2\circ\sigma\circ\mathbf{F}_1(x,y). \tag{4.2}$$

Here $\mathcal{T}$ is the shift layer (3.2) defined as before. Then, the 2D norm-DNN (4.2) will converge to the non-negative GS and the performance is quite similar as before. Let us illustrate this with two numerical examples. We fix $\beta=400$ and consider a box domain $\Omega=(-6,6)\times(-6,6)$ with equal-partitions in the following. The exact solution $\phi_g$ is obtained by traditional method. The training of norm-DNN is done by the Adam method with $tol=10^{-7}$, and a normalized Gaussian $\mathrm{e}^{-(\frac{x^2}{10}+\frac{y^2}{10})}/\sqrt{5\pi}$ is used for pre-training with 1000 iterations.

**Example 4.1.** (Harmonic oscillator potential) We first consider $V=\frac{1}{2}(x^2+y^2)$. The hyper-parameters are set as: $L\in\{1,2,3,4\}$, $W\in\{10,50,70\}$ and $N_x=Ny=64$. As in 1D case, $\sigma=\tanh$ is used in (4.2). Under $L=4$, $W=70$, we show the profiles of the solutions, pointwise errors and the training process of norm-DNN in Fig. 7. The errors (2.7) under different network architecture for the problem are shown in Table 7.

**Example 4.2.** (Optical lattice potential) Next, we consider $V=\frac{1}{2}(x^2+y^2)+\frac{5}{2}(\sin^2(\frac{\pi x}{4})+\sin^2(\frac{\pi y}{4}))$. Based on the study in Section 3.5, we change the activation function from 'tanh' to 'sin', i.e., $\sigma=\sin$ in (4.2). The hyper-parameters are set the same as above. The errors (2.7) are shown in Table 7. The profiles of solutions, pointwise errors and the training process under $L=4$, $W=70$ are shown in Fig. 8.

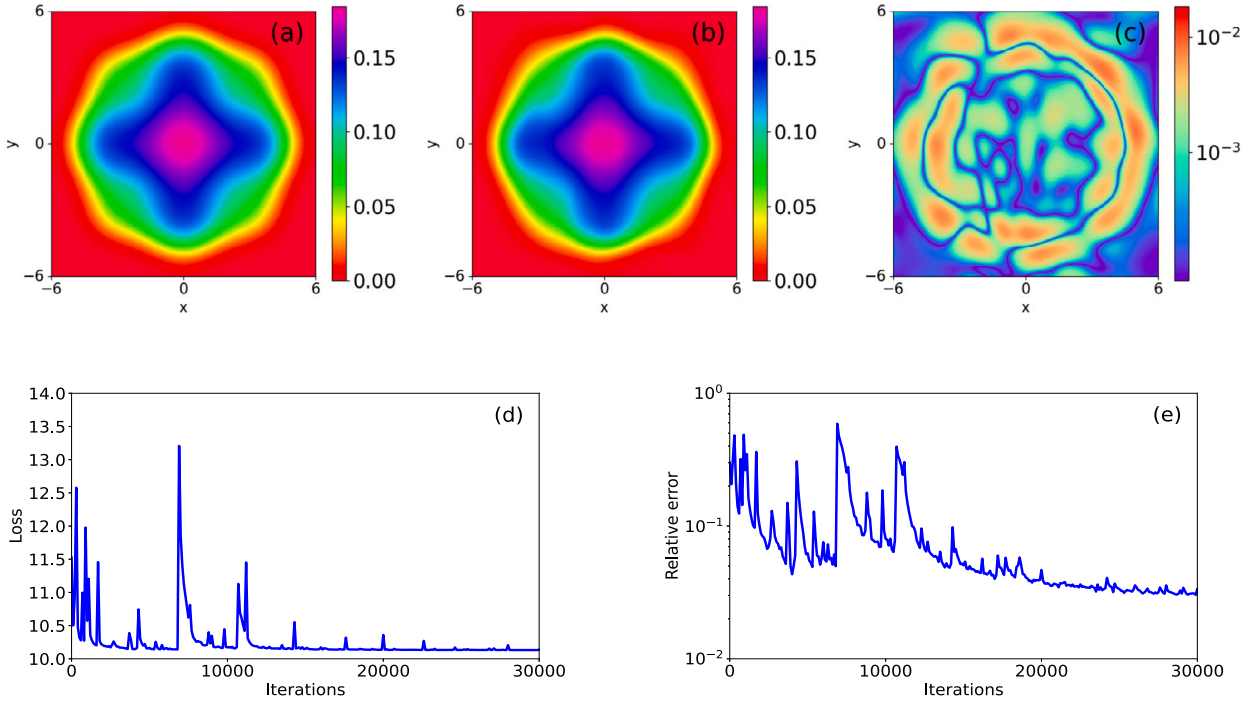


**Fig. 8.** Norm-DNN for 2D GS problem Example 4.2: exact GS (a); numerical solution of norm-DNN (b); pointwise error (c); loss (d) and error (2.7) (e) during the iterations.

From the above two numerical examples and the presented numerical results, we can observe the following. 1) The errors (2.7) of norm-DNNs drop significantly as the network architecture increases and when $L = 4$, $W = 70$, the relative error is about $2 \times 10^{-2}$. Correspondingly, the pointwise errors are kept below 1%. 2) Compared with the 1D case, we observe that the loss and error during the training process in Fig. 7(d,e) and Fig. 8(d,e) now exhibit some oscillations which cost more efforts to train than the 1D case, but they can still converge in the end. 3) The pre-training and the adjustment of the activation function when high frequency occurs, are also very effective in 2D. Overall, we conclude that norm-DNN can accurately capture the 2D GS, and all the strategies developed in 1D case work in high dimension.

One more issue for the high-dimensional problem is the grid points used to approximate the energy functional, e.g., the $(x_j, y_j)$ in (4.1). The equal-partition is not possible in a very high dimension and the usual way is to consider random sampling. To demonstrate the capacity of the proposed norm-DNN with random sampling, we consider the 2D GS problem in Example 4.1. Now, we randomly generate some points according to the uniform distribution in $\Omega$. The total number of the generated points is denoted as $N_b \in \{500, 1000\}$. Note that the number of points here is much less than that used in Example 4.1 to train norm-DNN, i.e., $N_b < 64 \times 64 = 4096$. The other setup remains the same as before, and we train the norm-DNN with $L = 4$, $W = 70$ for 10000 iterations. The solutions, pointwise errors and the changes of loss and relative error (2.7) along with the iterations are shown in Fig. 9.

When $N_b = 500$, we can clearly observe the pollution of noise in the numerical solution in Fig. 9(b). Nevertheless, norm-DNN in this case is still able to capture the general shape of GS. When the number of the training points is increased to $N_b = 1000$, the impact of random noise is largely reduced. Thus, norm-DNN with random sampling can work for the high-dimensional problem and the numerical solution can at least be qualitatively correct. At last, we remark that the randomness in sampling can lead to stronger oscillations in the loss and error during the training process, as shown in Fig. 9(d,e). The oscillations make it harder for the training to reach a stable state, and this is the reason why we set a fixed number of iterations for this test. Such issue may be relieved with a better sampling method, e.g., [59,42], or a better optimization method or stopping criterion, and this will be addressed in a future work.

Finally, we end this subsection by a three-dimensional (3D) test. The GS problem of a 3D BEC, i.e., $d = 3$ and $\mathbf{x} = (x, y, z)^\top \in \mathbb{R}^3$ in (1.1), by norm-DNN is then to consider the optimization:

$$\min_{\theta} \left\{ \text{Loss} = \frac{|\Omega|}{N_{3D}} \sum_{j=1}^{N_x} \sum_{k=1}^{N_y} \sum_{l=1}^{N_z} \left[ \frac{1}{2} |\nabla \phi_\theta(x_j, y_k, z_l)|^2 + V(x_j, y_k, z_l) |\phi_\theta(x_j, y_k, z_l)|^2 + \frac{1}{2} \beta |\phi_\theta(x_j, y_k, z_l)|^4 \right] \right\}$$

with $N_{3D} = N_x \times N_y \times N_z$ and $\phi_\theta(x, y, z) = \mathcal{N} \circ \mathcal{T} \circ \mathbf{F}_{L+1} \circ \sigma \circ \mathbf{F}_L \circ \sigma \circ \cdots \circ \mathbf{F}_2 \circ \sigma \circ \mathbf{F}_1(x, y, z)$. We fix $\beta = 200$ and set the computation domain $\Omega = (-4, 4) \times (-4, 4) \times (-4, 4)$ for $V = V_1 = \frac{1}{2}(x^2 + y^2 + z^2)$ and $\Omega = (-6, 6) \times (-6, 6) \times (-6, 6)$ for $V = V_2 = \frac{1}{2}(x^2 + y^2 + z^2) + \frac{5}{2}(\sin^2(\frac{\pi x}{4}) + \sin^2(\frac{\pi y}{4}) + \sin^2(\frac{\pi z}{4}))$. We use a normalized 3D Gaussian $e^{-(\frac{x^2}{10} + \frac{y^2}{10} + \frac{z^2}{10})}/(5\pi)^{3/4}$ for pre-training with 1000 iterations. A

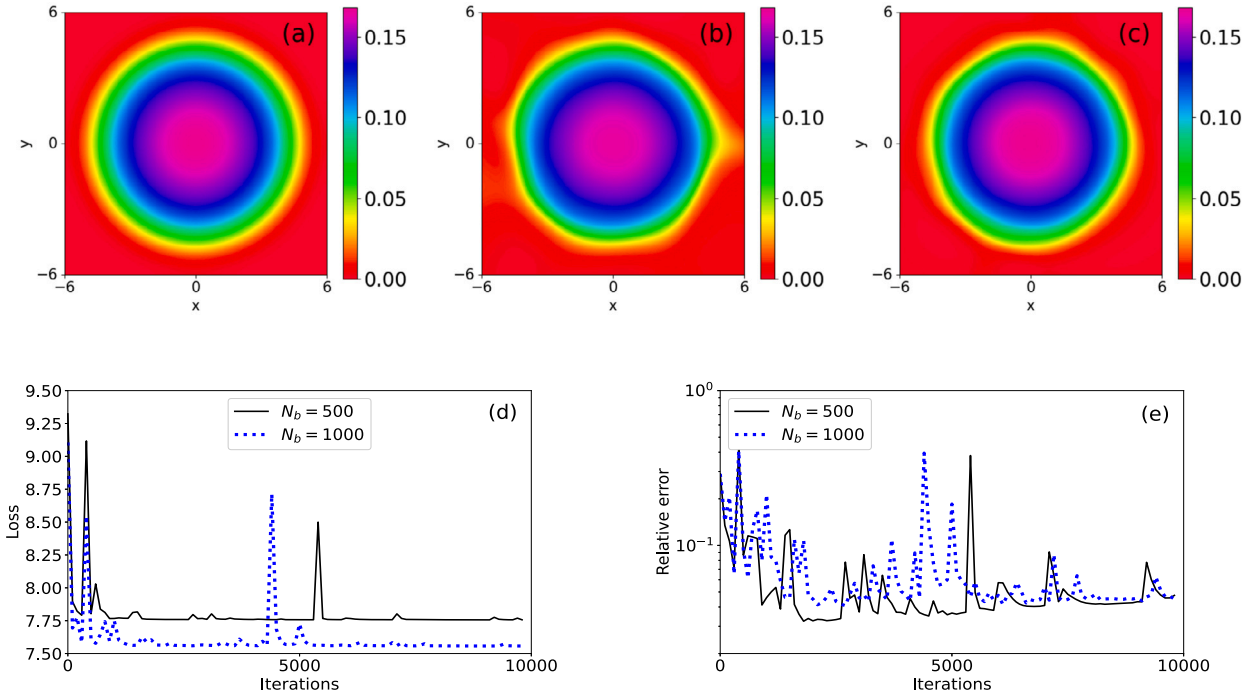


**Fig. 9.** norm-DNN with randomly selected training points for 2D GS problem: exact GS (a); numerical solution of norm-DNN with $N_b = 500$ (b) and $N_b = 1000$ (c); loss (d) and error (2.7) (e) during the iterations.

total number of the quadrature points $N_b = 10000 < 64 \times 64 \times 64 = 262144$ is randomly generated via the uniform distribution in $\Omega$. We train the norm-DNN with $L = 4$, $W = 100$ until the stopping condition $tol = 10^{-7}$ for the Adam method is met. In Fig. 10, we display the numerical solutions of the harmonic oscillator potential $V_1$ and the optical lattice potential $V_2$ in isosurface plot and also in contour plot, where the correct shape of 3D GS has been captured. This further demonstrates the potential of norm-DNN to solve high-dimensional GS problems.

**Remark 4.3.** We mention here the computational time (cputime) of GFDN in the previous examples. For the 1D problem in Section 3.4, the cputime is 1.2 s. However, for the 3D case, the cputime required by GFDN (912 s) is almost the same as the training time of norm-DNN (980 s). It might not be fair to compare norm-DNN with GFDN directly through the training time and the cputime. However, this certainly indicates that norm-DNN would become much more efficient for high-dimensional problems, which is the ultimate goal of norm-DNN. Besides, for norm-DNN, once the network is trained, it provides the solution value at any input $\mathbf{x}$ effortlessly, while GFDN produces the numerical solution only on the grid points.

### *4.2. Two-component BEC*

In this subsection, we consider a 1D two-component BEC with an internal atomic Josephson junction, which in dimensionless form reads as follows [7–9,57]:

$$\begin{aligned} i\frac{\partial \psi_1(x,t)}{\partial t} &= \Big[-\frac{1}{2}\partial_x^2 + V(x) + \beta_{11}|\psi_1(x,t)|^2 + \beta_{12}|\psi_2(x,t)|^2\Big]\psi_1(x,t) + \lambda\psi_2(x,t), \quad x \in \mathbb{R},\ t > 0, \\ i\frac{\partial \psi_2(x,t)}{\partial t} &= \Big[-\frac{1}{2}\partial_x^2 + V(x) + \beta_{21}|\psi_1(x,t)|^2 + \beta_{22}|\psi_2(x,t)|^2\Big]\psi_2(x,t) + \lambda\psi_1(x,t), \quad x \in \mathbb{R},\ t > 0, \end{aligned} \tag{4.3}$$

where $\Psi(x,t) = (\psi_1(x,t), \psi_2(x,t))^{\mathrm{T}}$ is the unknown vector-valued wave function, $V$ is the real-valued potential and $\beta_{11}, \beta_{12}(= \beta_{21}), \beta_{22}, \lambda \in \mathbb{R}$ are given constants. The two-component BEC (4.3) conserves the total (normalized) mass

$$M(\Psi(\cdot,t)) := \int_{\mathbb{R}} \big[|\psi_1(x,t)|^2 + |\psi_2(x,t)|^2\big]\, dx \equiv 1, \quad t \geq 0, \tag{4.4}$$

and the energy

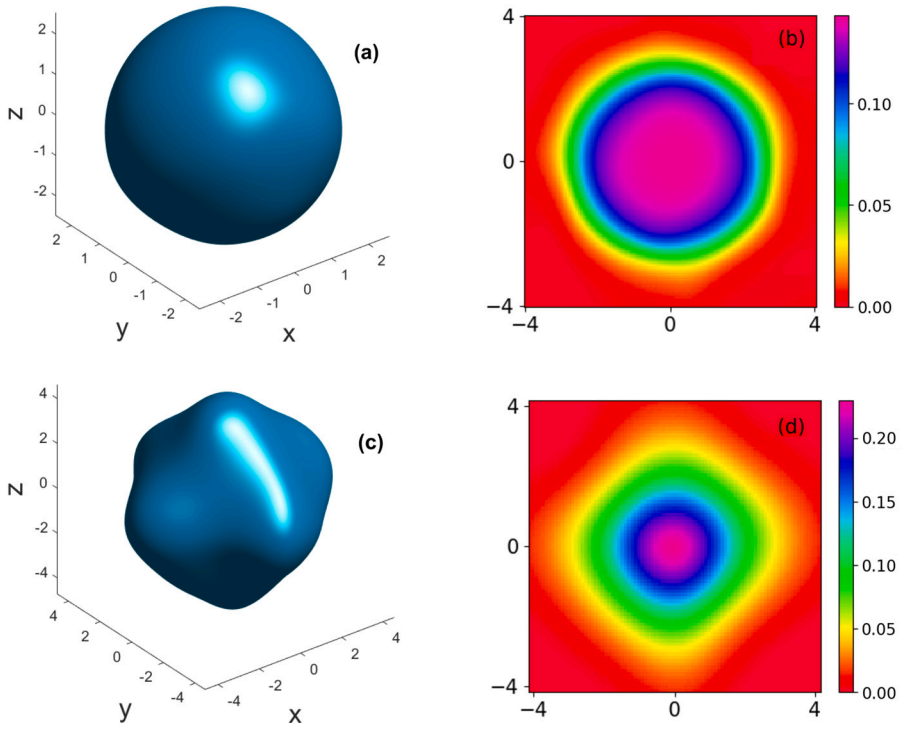


**Fig. 10.** Numerical solutions of norm-DNN for 3D GS problem: (a) $\phi_\theta$ and (b) $\phi_\theta$ at $z=0$ for the harmonic oscillator potential $V_1$; (c) $\phi_\theta$ and (d) $\phi_\theta$ at $z=0$ for the optical lattice potential $V_2$.

$$E(\Psi(\cdot,t)) := \int_{\mathbb{R}} \Big[\frac{1}{2}(|\partial_x\psi_1|^2+|\partial_x\psi_2|^2)+V(x)(|\psi_1|^2+|\psi_2|^2)+\frac{1}{2}\beta_{11}|\psi_1|^4+\frac{1}{2}\beta_{22}|\psi_2|^4 + \beta_{12}|\psi_1|^2|\psi_2|^2+2\lambda\cdot\mathrm{Re}(\psi_1\overline{\psi}_2)\Big]dx \equiv E(\Psi(\cdot,0)), \quad t\ge 0.$$

The theoretical results in [7–9] indicate that the GS can be real-valued, and thus the GS problem for (4.3) can be considered as [7,8]: find $\Phi_g=(\phi_{1,g},\phi_{2,g})^\top\in S$ such that

$$\widetilde{E}(\Phi_g)=\min_{\Phi\in S}\widetilde{E}(\Phi),\ \text{with } S=\left\{\Phi=(\phi_1,\phi_2)^\top\ :\ \int_{\mathbb{R}}\left[|\phi_1(x)|^2+|\phi_2(x)|^2\right]dx=1,\ \Phi:\mathbb{R}\to\mathbb{R}^2\right\}, \tag{4.5}$$

where

$$\widetilde{E}(\Phi) := \int_{\mathbb{R}} \Big[\frac{1}{2}(|\phi_1'|^2+|\phi_2'|^2)+V(x)(|\phi_1|^2+|\phi_2|^2)+\frac{1}{2}\beta_{11}|\phi_1|^4+\frac{1}{2}\beta_{22}|\phi_2|^4+\beta_{12}|\phi_1|^2|\phi_2|^2+2\lambda\phi_1\phi_2\Big]dx.$$

For the related theoretical results of (4.5), we refer the readers to [9]. Here let us focus on the computation of the GS $\Phi_g$ by extending the proposed norm-DNN method. Based on our previous studies of the single component BEC, we consider the norm-DNN in the same form as before:

$$\Phi_\theta(x)=\big(\phi_{1,\theta}(x),\phi_{2,\theta}(x)\big)^\top=\mathcal{N}\circ\mathcal{T}\circ\mathbf{F}_{L+1}\circ\sigma\circ\mathbf{F}_L\circ\sigma\circ\cdots\circ\mathbf{F}_2\circ\sigma\circ\mathbf{F}_1(x). \tag{4.6}$$

Note the only difference is now we set two neurons in the output layer, i.e., $n_{L+1}=2$, to represent the approximations for $\phi_1$ and $\phi_2$. To satisfy the mass constraint (4.4), we modify the normalization layer $\mathcal{N}$ as:

$$\mathcal{N}(\mathbf{y})=\frac{\mathbf{y}}{\sqrt{||y_1||_2^2+||y_2||_2^2}},\quad \mathbf{y}=\mathbf{y}(x)=(y_1(x),y_2(x))^\top:\mathbb{R}\to\mathbb{R}^2,$$

and set the shift layer $\mathcal{T}$ as

$$\mathcal{T}(\mathbf{y}):=\mathcal{T}\begin{pmatrix}y_1\\y_2\end{pmatrix}=\begin{pmatrix}y_1-\min_x(y_1)\\y_2-\min_x(y_2)\end{pmatrix},\quad \mathbf{y}=\mathbf{y}(x)=(y_1(x),y_2(x))^\top:\mathbb{R}\to\mathbb{R}^2.$$

**Table 8**
Error (2.7), number of iterations and training time of norm-DNN for two-component GS problem. Here $\text{Error}_1$ and $\text{Error}_2$ represent the relative error (2.7) for $\phi_1$ and $\phi_2$, respectively.

| | Iterations | Time | $\text{Error}_1$ | $\text{Error}_2$ |
|---|---|---|---|---|
| $\beta = 300$ | 6800 | 49 s | 6.74E-3 | 6.38E-3 |
| $\beta = 800$ | 5400 | 41 s | 5.17E-3 | 4.95E-3 |

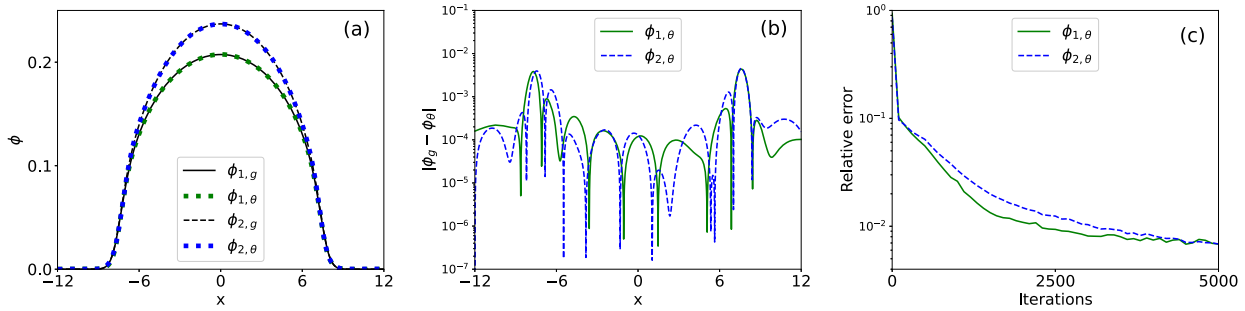


**Fig. 11.** Norm-DNN for two-component GS problem with $\beta = 300$. The 1st to 3rd subfigures represent in sequence: profile of solution, pointwise error, training process of error (2.7).

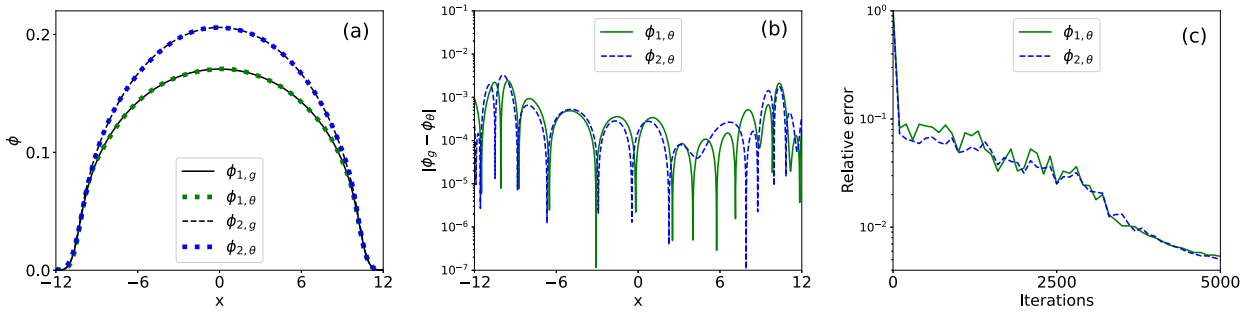


**Fig. 12.** Norm-DNN for two-component GS problem with $\beta = 800$. The 1st to 3rd subfigures represent in sequence: profile of solution, pointwise error, training process of error (2.7).

Now we consider a numerical example to illustrate the performance of (4.6) for the two-component GS problem. Choose $\Omega = (-12, 12)$, $V(x) = \frac{1}{2}x^2$, $\beta_{11} = \beta$, $\beta_{12} = 0.94\beta$, $\beta_{22} = 0.97\beta$ with $\beta = 300$ or 800, $\lambda = -1$. The exact $\Phi_g$ can be obtained by the extended GFDN scheme from [9]. We use the same Gaussian $\phi_0$ as the pre-training to initialize $\phi_1$ and $\phi_2$. The hyper-parameters are set as $L = 4$, $W = 70$, $tol = 10^{-7}$, $N_x = 128$. The relative error (2.7), number of iterations and training time are shown in Table 8. The exact and numerical solutions of norm-DNNs for $\phi_1$ and $\phi_2$, pointwise errors, the convergence process of relative error (2.7)are shown in Fig. 11 and Fig. 12.

From the numerical results, we can see that norm-DNNs can accurately and quickly get the GS of the two-component BEC problem, with only the minor cost of adding an extra neuron in the output layer. This further demonstrates the effectiveness of the proposed norm-DNN for solving GS problems. We remark that another common way to fit multiple target functions is to use multiple networks to fit each function individually. For the two-component GS problem, using two different networks would cause difficulty to impose the normalization layer to meet (4.4) and this will lead to additional computational costs.

### *4.3. First excited state*

The other eigenfunctions of (1.6) with energy larger than $E_g$ are referred as excited states. In addition to GS, the first excited state (FES) of the BEC is also widely concerned [10]. It can be characterized as the minimizer of the energy functional in the orthogonal space of GS under the mass constraint, i.e.,

$$\phi_{1\text{st}} \in \arg\min\{E(\phi) \ : \ \|\phi\|_2 = 1, \ <\phi, \phi_g> = 0\}, \tag{4.7}$$

with $<\cdot,\cdot>$ denotes the inner product in $L^2(\mathbb{R}^d)$. Now we present the special strategy for norm-DNN to compute it. As given in Section 2.1, the FES $\phi_{1\text{st}}$ is orthogonal to $\phi_g$ in $L^2$. When the potential $V(x)$ is an even function, $\phi_{1\text{st}}$ is in fact known to be an odd function [9] that minimizes the energy on the unit sphere of $L^2$. The traditional GFDN method therefore will choose an odd initial

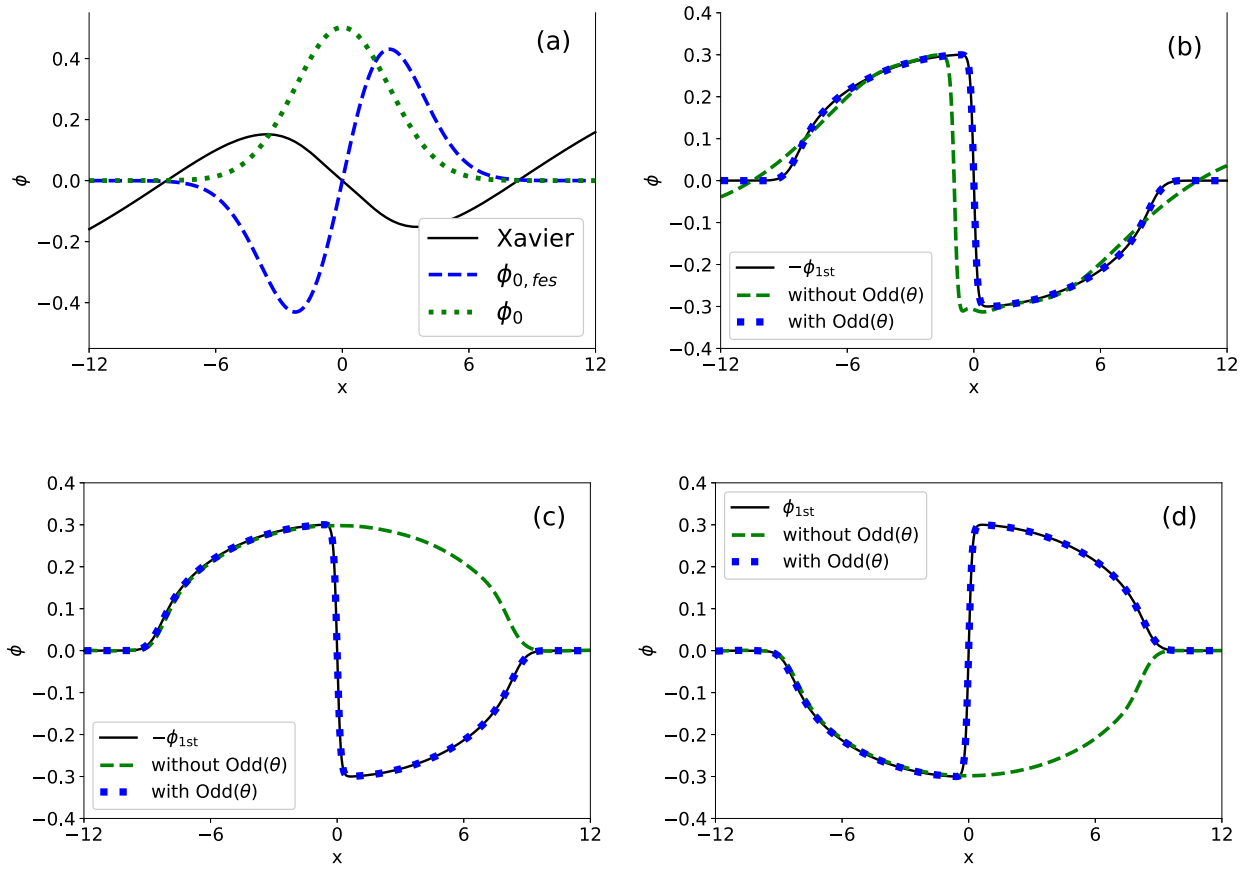


**Fig. 13.** FES problem: (a) initial profile of norm-DNN from pre-training or Xavier method [28]; (b) norm-DNN $\phi_\theta$ with Xavier method; (c) norm-DNN $\phi_\theta$ pre-trained from $\phi_0$; and (d) norm-DNN $\phi_\theta$ pre-trained from $\phi_{0,fes}$.

guess, e.g., $\phi_{0,fes}(x) := \sqrt{2}xe^{-\frac{x^2}{10}}/(5^3\pi)^{1/4}$, and then the flow will stay in the correct functional space to reach $\phi_{1st}$. Unfortunately, our norm-DNN cannot preserve the parity of the initial data to the end of training. This fact has already been shown in Fig. 4(b), where training the norm-DNN from an even function $\phi_0$ may yield an odd numerical solution. On the other hand, the training from a normalized odd function $\phi_{0,fes}$ may yield an even solution, as will be shown in Fig. 13(b,c,d).

To efficiently and accurately fit the FES, we propose the following two modifications for the norm-DNN approach:

- Unlike GS, FES does not satisfy the non-negative property, and so we do not need the shift layer $\mathcal{T}$ (3.2) in (3.3). Therefore for the 1D FES problem, the norm-DNN simply reads:

$$\phi_\theta(x) = \mathcal{N} \circ \mathbf{F}_{L+1} \circ \sigma \circ \mathbf{F}_L \circ \sigma \circ \cdots \circ \mathbf{F}_2 \circ \sigma \circ \mathbf{F}_1(x). \tag{4.8}$$

- We add a regularization term in the loss to restrict $\phi_\theta(x)$ being odd in space, and so we solve the following minimization problem:

$$\min_\theta \{\text{Loss}(\theta) + \text{Odd}(\theta)\}, \qquad \text{Odd}(\theta) := \frac{\delta}{N_o}\sum_{j=1}^{N_o} |\phi_\theta(x_j^o) + \phi_\theta(-x_j^o)|^2, \tag{4.9}$$

with a wight $\delta > 0$ and the training points $\{x_j^o\}_{j=1}^{N_o} \subset \Omega \cap \mathbb{R}^+$ for the odd-function restriction.

Now let us fix $V(x) = \frac{1}{2}x^2$, $\beta = 400$ for numerical investigations. The hyper-parameters are set as $L = 4$, $W = 70$, $tol = 10^{-6}$, $N_x = 512$. The weight in (4.9) is taken as $\delta = 100$ and the points $x_o^j$ are taken as the grid points in $[0, 12]$. In Table 9, we present the number of iterations, computational time, the relative function error (2.7) and the error of the numerical energy $E(\phi_\theta)$ of norm-DNN for the FES problem. To address the necessity of the regularization in (4.9), we have presented the numerical results of norm-DNN with or without using Odd($\theta$) for comparison. Moreover, the Xavier method [28] or the pre-training with the Gaussian $\phi_0$ or the odd function $\phi_{0,fes}$ has been used for the initialization. The profiles of the solutions are shown in Fig. 13.

From the numerical results, we have the following findings. 1) Norm-DNN with the regularization Odd($\theta$) provides accurate and efficient approximation to the FES, and the energy error $|E(\phi_{1st}) - E(\phi_\theta)|$ can reach $10^{-4}$. 2) The random initialization for (4.9)

**Table 9**
Number of iterations, computational time, relative error (2.7), the error of numerical energy $E(\phi_\theta)$ of norm-DNN for FES problem (exact FES energy $E(\phi_{1\text{st}}) = 22.0777$).

| | Initialization | Iterations | Time | Error (2.7) | $\lvert E(\phi_{1\text{st}}) - E(\phi_\theta)\rvert$ |
|---|---|---|---|---|---|
| with Odd | $\phi_{0,fes}$ | 2400 | 19 s | 2.68E-3 | 1.00E-4 |
| | $\phi_0$ | 2100 | 17 s | 2.07E-3 | 1.00E-4 |
| | Xavier [28] | 15500 | 118 s | 2.90E-3 | 1.00E-4 |
| without Odd | $\phi_{0,fes}$ | 2300 | 26 s | 1.41E+0 | 7.28E-1 |
| | $\phi_0$ | 4600 | 34 s | 1.41E+0 | 7.28E-1 |
| | Xavier [28] | 2400 | 16 s | 1.93E+0 | 1.15E-1 |

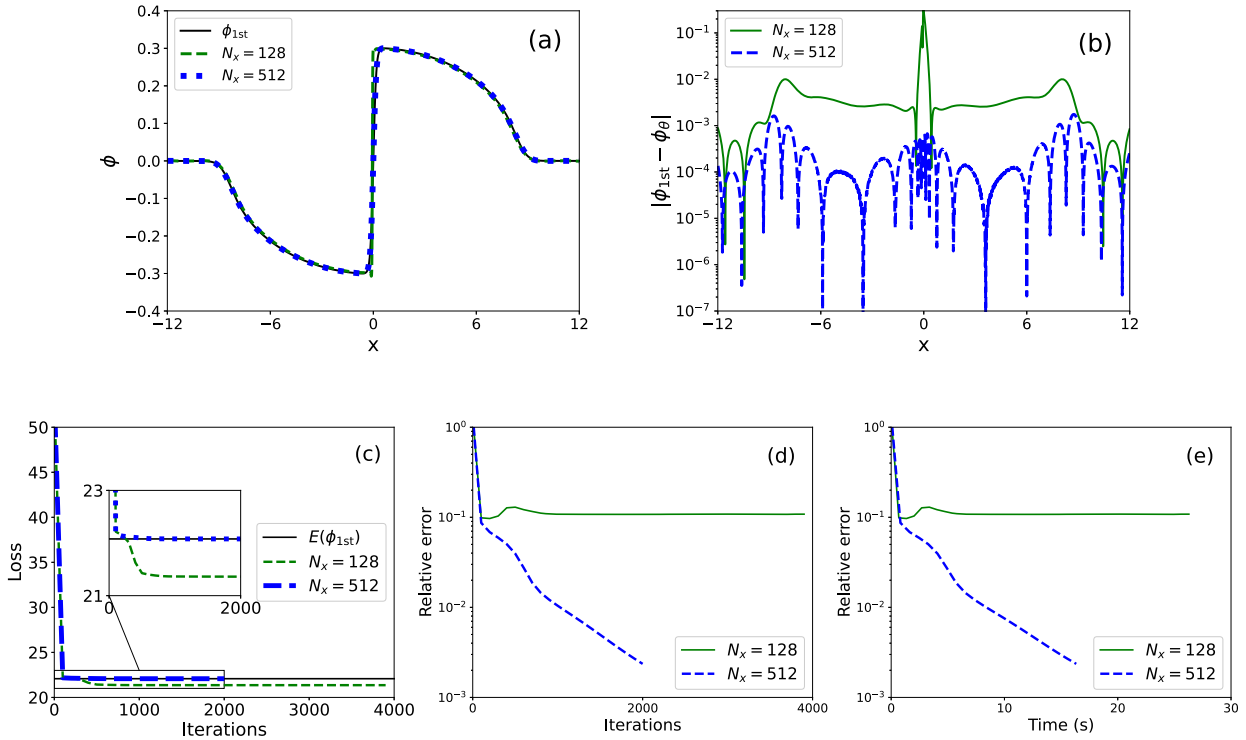


**Fig. 14.** Norm-DNN with different $N_x$ for FES problem: (a) exact solution and numerical solution of norm-DNN; (b) pointwise error; (c,d,e) loss and error (2.7) during the iterations or computational time.

**Table 10**
The relative error (2.7) and training time of norm-DNN with different $N_x$ for FES problem.

| $N_x$ | 64 | 128 | 256 | 512 |
|---|---|---|---|---|
| Error(Time) | 1.34E-1(15 s) | 1.08E-1(26 s) | 1.03E-1(60 s) | 2.07E-3(17 s) |

which at the beginning is far away from the FES, can still converge to the FES $\phi_{1\text{st}}$ and approximate it accurately. This indicates the robustness of the proposed approach for FES problem. 3) Pre-training with the Gaussian $\phi_0$ or the odd function $\phi_{0,fes}$ can significantly accelerate the convergence. The difference between the two is subtle. 4) Without using Odd($\theta$), norm-DNN could produce the wrong solution despite of the used initialization. It turns to GS or some other states instead of FES.

One more thing that we would like to address here is the number of spatial grid points used for FES. The previous experiments for the GS Problem 3.4 use $N_x = 128$, which is enough for the training to converge quickly and the error can reach the order of $10^{-3}$. Here for the FES problem, we need to take more grid points to accurately capture the sudden jump in $\phi_{1\text{st}}$ at the origin. To illustrate this issue, we test the performance of the training for $N_x \in \{64, 128, 256, 512\}$. The other parameters are fixed as before and the Gaussian $\phi_0$ for pre-training is used. The relative error (2.7) and training time of norm-DNNs are given in Table 10. The piecewise errors and the changes of loss and relative error (2.7) along with the iterations and time are shown in Fig. 14. The numerical results show that norm-DNN with $N_x = 512$ gives the fastest convergence rate and best accuracy for computing FES. When there is insufficient training data, the approximation could be poor especially at the jump, and overfittings may occur as shown by the green line in Fig. 14(d,e).

**Table 11**
Error (2.7) of norm-DNN on grid points of parameter space.

| $\beta$ \ $\gamma$ | 0.5 | 0.6 | 0.7 | 0.8 | 0.9 | 1.0 |
|---|---|---|---|---|---|---|
| 300 | 7.46E-3 | 8.18E-3 | 7.69E-3 | 9.13E-3 | 1.01E-2 | 8.59E-3 |
| 310 | 9.02E-3 | 7.89E-3 | 7.45E-3 | 8.83E-3 | 8.81E-3 | 8.99E-3 |
| 320 | 9.52E-3 | 9.00E-3 | 6.84E-3 | 9.43E-3 | 9.91E-3 | 1.07E-2 |
| 330 | 1.14E-2 | 8.62E-3 | 8.90E-3 | 6.85E-3 | 1.08E-2 | 9.56E-3 |
| 340 | 1.05E-2 | 9.00E-3 | 8.40E-3 | 6.98E-3 | 9.02E-3 | 9.87E-3 |
| 350 | 1.20E-2 | 8.32E-3 | 9.01E-3 | 8.15E-3 | 9.03E-3 | 9.01E-3 |
| 360 | 1.29E-2 | 9.04E-3 | 8.12E-3 | 6.67E-3 | 9.15E-3 | 1.07E-2 |
| 370 | 1.34E-2 | 9.37E-3 | 8.20E-3 | 8.08E-3 | 8.33E-3 | 9.87E-3 |
| 380 | 1.28E-2 | 9.43E-3 | 8.68E-3 | 9.74E-3 | 7.37E-3 | 9.72E-3 |
| 390 | 1.25E-2 | 1.10E-2 | 8.31E-3 | 9.36E-3 | 7.92E-3 | 9.78E-3 |
| 400 | 1.20E-2 | 1.11E-2 | 7.76E-3 | 9.67E-3 | 7.71E-3 | 7.64E-3 |

## 5. Generalization and comparison

The previous experiments have investigated the norm-DNN for approximating a single GS of (1.1) under a fixed set of physical parameters, e.g., $\beta$ and $\gamma_j$ in $V$. Mathematically, the GS $\phi_g$ can be viewed as a function depending on these physical parameters. Thus for applications that concern parameters from a certain range, it would be more interesting and useful to have a fully trained norm-DNN in parameters space. Such trained norm-DNN will then be able to directly provide a valid numerical approximation of GS for any parameters within the range, which makes it very simple and efficient for applications. This is considered as the parameter generalization of norm-DNN, which we shall study in this section.

### 5.1. Generalization

The traditional approaches for computing GS are all designed under fixed parameters. It means that one needs to specify the values of all the physical parameters in order to run the numerical scheme to get the specific solution. Whenever the value of one parameter is changed, one will have to run the scheme once again for the new solution. This could certainly be very costly and tedious for applications. The DNNs in classical machine learning are known to have strong parameter generalization ability. We now propose such parameter generalization for norm-DNN to compute GS within a range of parameters and investigate its performance. Let us consider the 1D GS problem as the illustrative example for simplicity. We take the harmonic oscillator potential $V(x) = \gamma x^2$. Then, the concerned physical parameters in the model (3.1) are $\gamma$ and $\beta$. The parameter generalization procedure is presented in Algorithm 1.

**Algorithm 1** Parameter generalization for norm-DNN.

**Input:** Domain for parameters $P = \{(\gamma, \beta)\}$ and physical domain $\Omega$
**Output:** Fully trained norm-DNN $f_\Theta(\gamma, \beta) \approx \phi_g$ for any $(\gamma, \beta) \in P$

1: Sample points for training: $P_s = \{(\gamma_i, \beta_j)\} \subset P$, $\{x_k\}_{k=1}^{N_x} \subset \Omega$
2: For each $(\gamma_i, \beta_j) \in P_s$, compute the norm-DNN from (3.1) and denote it as $\phi_\theta^{i,j}(x)$
3: Give spatial points $\{x_l\}_{l=1}^{N_t} \subset \Omega$ where the value of $\phi_g$ is of interest, then evaluate $\phi_\theta^{i,j}$ at $\{x_l\}_{l=1}^{N_t}$
4: Set $\phi_\theta^{i,j}(x_l)$ as the label and do supervised training: generate a FCNN $f_\Theta(\gamma, \beta) = (f_\Theta^1, \ldots, f_\Theta^{N_t})^\top$, i.e., (2.2) with $n_{L+1} = N_t$, $n_0 = 2$; train it by considering

$$\min_\Theta \sum_{P_s} \sum_{l=1}^{N_t} |\phi_\theta^{i,j}(x_l) - f_\Theta^l(\gamma_i, \beta_j)|^2 \tag{5.1}$$

**Remark 5.1.** The output of Algorithm 1 takes discrete values in the physical domain $\Omega$, i.e., $f_\Theta \in \mathbb{R}^{N_t}$ for every $(\gamma, \beta)$. It approximates $\phi_g$ at the specified points $\{x_l\}_{l=1}^{N_t}$. These points can be chosen or changed based on the practical need. If changed, one can just repeat the steps 3-4 of Algorithm 1 with the new $x_l$ to get the FCNN. Besides, combining the parameter $\theta$ and spatial variable $x$ optimization, an object would be training $f_\Theta$ as a function of the trio $(\gamma, \beta, x)$, which via the framework of PINO [45] or DeepONet [40] is possible and may enhance the performance, and this will be considered in a future work.

To investigate the performance of Algorithm 1, we consider a precise experiment. Fix the physical domain $\Omega = (-12, 12)$ and choose the parameter domain as $P = \{(\gamma, \beta) : 0.5 \le \gamma \le 1, 300 \le \beta \le 400\}$. A $11 \times 11$ equally spaced grid of $P$ is used to train the norm-DNN in the parameter space. Four test points $\{0.725, 0.825\} \times \{355, 365\}$ outside the grid points of $P$ are used to compute the error and measure the generalization ability. We take $L = 4$, $W = 70$ for both of the networks, i.e., norm-DNN and FCNN in Algorithm 1. The other hyper-parameters are set as: $N_x = N_t = 128$ and $tol = 10^{-5}$. The minimization at the supervised learning step in Algorithm 1 is done by Adam method with fixed 10000 iterations. In Table 11, we first show the error of norm-DNN for computing the GS with fixed parameters at some training grid points in $P$. On the four test parameter points, the relative error (2.7) (norm computed over the grid with $N_x$) are shown in Table 12. The profiles of the solutions at the test points are shown in Fig. 15.

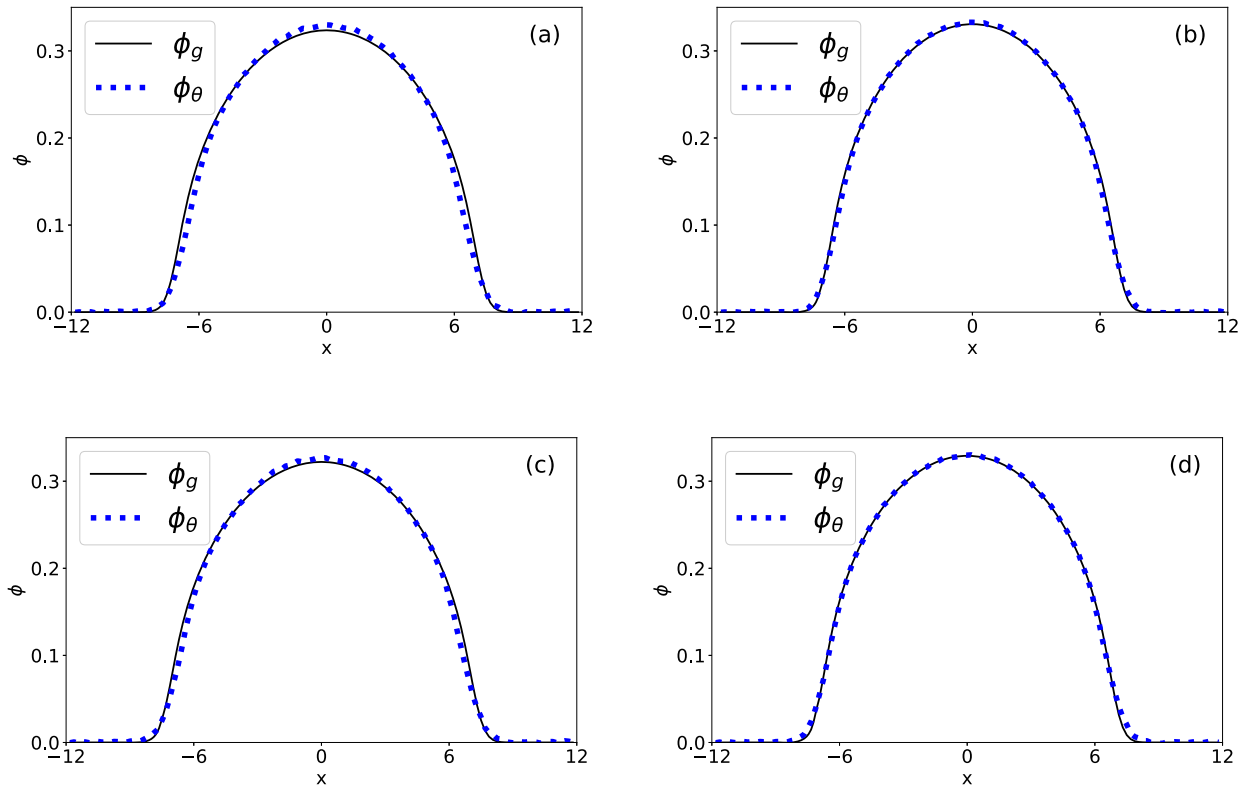


**Fig. 15.** Profiles of exact solution and numerical solution from norm-DNN/FCNN at the four test points in parameter space: (a) $\beta = 355$, $\gamma = 0.725$; (b) $\beta = 355$, $\gamma = 0.825$; (c) $\beta = 365$, $\gamma = 0.725$; (d) $\beta = 365$, $\gamma = 0.825$.

**Table 12**
Error (2.7) of FCNN on the four test points in parameter space.

| Error (2.7) | $\beta = 355$ | $\beta = 365$ |
|---|---|---|
| $\gamma = 0.725$ | 4.19E-2 | 3.26E-2 |
| $\gamma = 0.825$ | 1.84E-2 | 1.51E-2 |

From the numerical results, it can be observed that for different parameters, norm-DNNs can get all the GS with relative error at about 1%, which to some extent indicates the robustness of norm-DNN in parameters. For the parameters outside the grid points for training, the FCNN is also able to offer the approximation with the same magnitude of accuracy. This shows the validity of the fully trained norm-DNN and justifies the generalization ability of norm-DNN.

### 5.2. Comparison with traditional DNN

To further clarify the efficiency, we now carry out some comparisons between the proposed norm-DNN approach and the traditional DNN with supervised learning approach (SL-DNN) for the GS problem.

Firstly, we consider the approximation of GS under fixed parameters. For the traditional SL-DNN, we first obtain the GS $\phi_g$ by the GFDN method (2.1), and then we use its function values at the spatial grid points as the labeled data for supervised learning: generate a FCNN $f_\theta(x)$ as in (2.2) to solve

$$\min_\theta \sum_{j=1}^{N_x} |\phi_g(x_j) - f_\theta(x_j)|^2.$$

With the obtained SL-DNN $f_\theta(x)$, we compute its relative error (2.7) as before, and we compare it with the result from norm-DNN. The following four examples from previous studies will be used for numerical experiments:

(a) 1D problem with $\Omega = (-12, 12)$, $\beta = 400$, $V(x) = \frac{1}{2}x^2$;
(b) 1D problem with $\Omega = (-12, 12)$, $\beta = 400$, $V(x) = \frac{1}{2}x^2 + 25\sin^2(\frac{\pi x}{4})$;
(c) 2D problem with $\Omega = (-6, 6) \times (-6, 6)$, $\beta = 400$, $V(x, y) = \frac{1}{2}(x^2 + y^2)$;

**Table 13**
Error (2.7) of supervised learning DNN and norm-DNN for fixed-parameter experiments (a)-(d).

| Experiments | SL-DNN | norm-DNN |
|---|---|---|
| (a) | 1.47E-2 | 4.57E-3 |
| (b) | 2.74E-2 | 1.51E-2 |
| (c) | 5.81E-2 | 2.75E-2 |
| (d) | 4.61E-2 | 3.10E-2 |

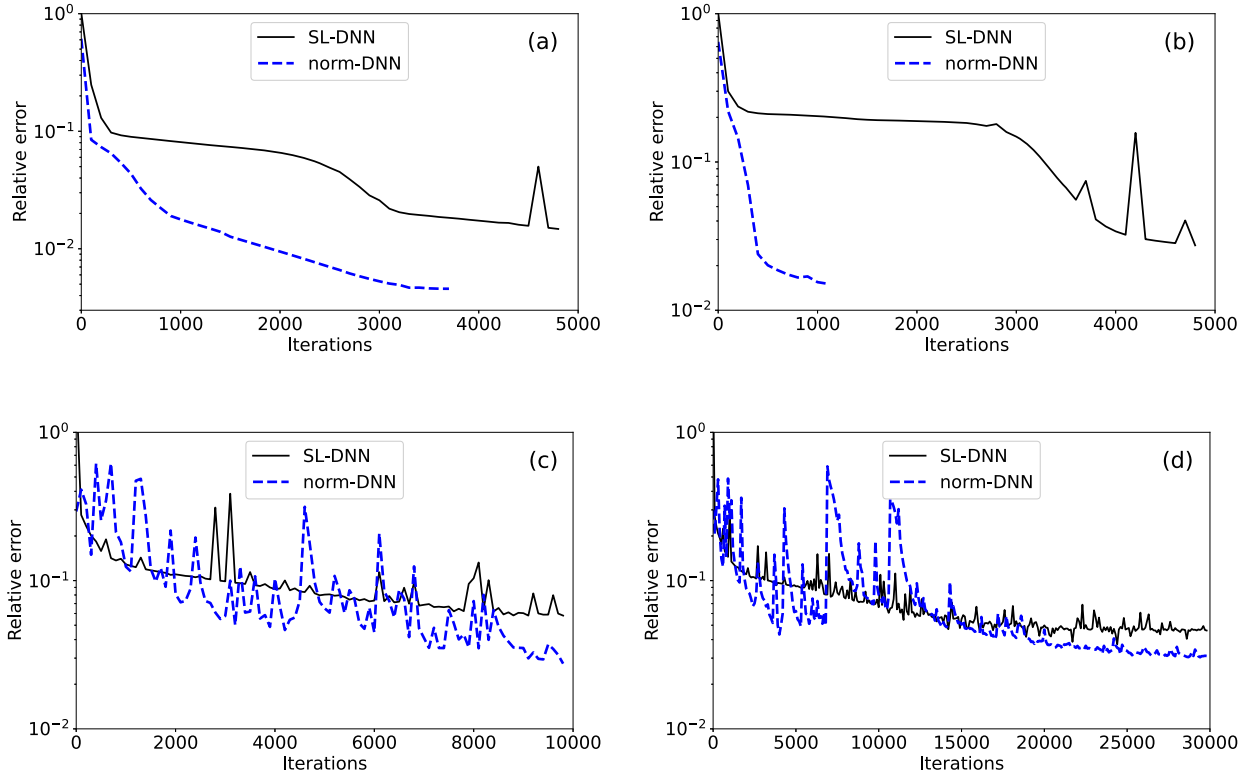


**Fig. 16.** Error (2.7) of the SL-DNN and norm-DNN during the iterations for the fixed-parameter experiments (a)-(d).

(d) 2D problem with $\Omega = (-6,6)\times(-6,6)$, $\beta = 400$, $V(x,y) = \frac{1}{2}(x^2+y^2) + \frac{5}{2}(\sin^2(\frac{\pi x}{4}) + \sin^2(\frac{\pi y}{4}))$.

The hyper-parameters are set the same as in Section 5.1. Then for the experiments (a)-(d), the corresponding relative error (2.7) and the training process are shown in Table 13 and Fig. 16. It can be seen from the numerical results that norm-DNN is more accurate than the supervised learning DNN. The training process of norm-DNN is at least as fast as that of the SL-DNN. Note that the SL-DNN requires the exact solution for training data, while norm-DNN is completely unsupervised and does not require any data points from the unknown. This shows the significant advantage of norm-DNN.

We also compare the parameter generalization of norm-DNN with the SL-DNN. Consider the numerical example from Section 5.1 with the same computational setup. The procedure of the SL-DNN method in this case goes similarly as (5.1) but with the labeled data now obtained from GFDN for each parameter $(\gamma_i, \beta_j)$. With the four test points: $\{0.725, 0.825\}\times\{355, 365\}$ in the parameter space, we quantify the accuracy by computing

$$\text{error} = \left\|\phi_\theta^{\text{test}} - \phi_g^{\text{test}}\right\|_F / \left\|\phi_g^{\text{test}}\right\|_F, \tag{5.2}$$

where $\phi_g^{\text{test}}, \phi_\theta^{\text{test}} \in \mathbb{R}^{4\times N_x}$ are the augmented exact and numerical GS solutions on the four test points and $\|\cdot\|_F$ denotes the Frobenius norm. The loss (5.1) and relative error (5.2) are shown in Fig. 17. Clearly the results show that for the parameter generalization problem, norm-DNN without data points can achieve the same accuracy and efficiency as the SL-DNN.

## 6. Conclusion

The ground state (GS) problem of the Gross-Pitaevskii model for Bose-Einstein condensate is a constraint optimization in functional space. In this paper, we applied the deep neural network (DNN) technique to numerically solve the problem with the intention to

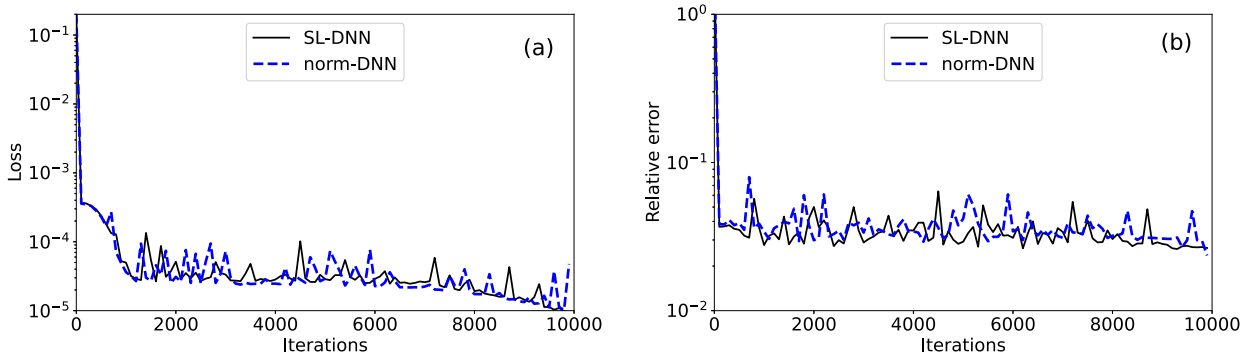


**Fig. 17.** Loss (5.1) and error (5.2) during the iterations for parameter generalization problem.

overcome possible curse-of-dimensionality and parameter generalization issues. In order to satisfy the mass constraint and guide the learning to GS, we proposed a normalized DNN (norm-DNN) by introducing a normalization layer and a shift layer into the traditional DNN. The norm-DNN converts the problem into a unconstrained optimization in finite dimensional parameter space, where GS as well as the excited states can be efficiently computed via unsupervised learning. Experiments from simple one-dimensional tests to the high-dimensional or multi-component cases have been done. Practical strategies for the choice of activation function, initialization and regularization of loss have been suggested under specific physical setups. The extensive numerical results illustrated the effectiveness and efficiency of the proposed norm-DNN approach. The generation of norm-DNN in the domain for physical parameters were also investigated, and comparisons were made with existing supervised learning. Of course, the truly meaningful and challenging problem is the solution of the many-body Schrödinger equation. Nevertheless, the findings in this paper show a promising perspective of norm-DNN in this direction, and this will be our future task.

## CRediT authorship contribution statement

**Weizhu Bao:** Supervision. **Zhipeng Chang:** Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Resources, Project administration, Methodology, Investigation, Funding acquisition, Formal analysis, Data curation, Conceptualization. **Xiaofei Zhao:** Supervision.

## Declaration of competing interest

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

## Acknowledgements

W. Bao is supported by the Ministry of Education of Singapore under its AcRF Tier 2 funding MOE-T2EP20122-0002 (A-8000962-00-00). Z. Chang and X. Zhao are partially supported by the National Natural Science Foundation of China 12271413 and the Natural Science Foundation of Hubei Province No. 2019CFA007.

## Data availability

Data will be made available on request.

## References

[1] R. Altmann, P. Henning, D. Peterseim, The $J$-method for the Gross-Pitaevskii eigenvalue problem, Numer. Math. 148 (2021) 575–610.
[2] M.H. Anderson, J.R. Ensher, M.R. Matthewa, C.E. Wieman, E.A. Cornell, Observation of Bose-Einstein condensation in a dilute atomic vapor, Science 269 (1995) 198–201.
[3] X. Antoine, W. Bao, C. Besse, Computational methods for the dynamics of the nonlinear Schrödinger/Gross-Pitaevskii equations, Comput. Phys. Commun. 184 (2013) 2621–2633.
[4] X. Antoine, A. Levittb, 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) 92–109.
[5] X. Bai, 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, V. Sethuraman, Bayesian optimization of Bose-Einstein condensates, Sci. Rep. 11 (2021) 5054.
[7] W. Bao, Analysis and efficient computation for the dynamics of two-component Bose-Einstein condensates: stationary and time dependent Gross-Pitaevskii equations, Contemp. Math. 473 (2008) 1–26.
[8] W. Bao, Y. Cai, Ground states of two-component Bose-Einstein condensates with an internal atomic Josephson junction, East Asian J. Appl. Math. 1 (2010) 49–81.
[9] W. Bao, Y. Cai, Mathematical theory and numerical methods for Bose-Einstein condensation, Kinet. Relat. Models 6 (2013) 1–135.

[10] W. Bao, I. Chern, F.Y. Lim, Efficient and spectrally accurate numerical methods for computing ground and first excited states in Bose-Einstein condensates, J. Comput. Phys. 219 (2006) 836–854.
[11] W. Bao, Q. Du, Computing the ground state solution of Bose-Einstein condensates by a normalized gradient flow, SIAM J. Sci. Comput. 25 (2004) 1674–1697.
[12] G. Bao, D. Wang, B. Zou, WANCO: weak adversarial networks for constrained optimization problems, arXiv preprint arXiv:2407.03647, 2024.
[13] A.R. Barron, Universal approximation bounds for superpositions of a sigmoidal function, IEEE Trans. Inf. Theory 39 (1993) 930–945.
[14] Y. Bengio, A. Courville, P. Vincent, Representation learning: a review and new perspectives, IEEE Trans. Pattern Anal. Mach. Intell. 35 (2013) 1798–1828.
[15] S.N. Bose, Plancks Gesetz und Lichtquantenhypothese, Z. Phys. 26 (1924) 178–181.
[16] E. Cancès, R. Chakir, Y. Maday, Numerical analysis of nonlinear eigenvalue problems, J. Sci. Comput. 45 (2010) 90–117.
[17] Z. Chang, K. Li, X. Xiang, X. Zou, High order deep neural network for solving high frequency partial differential equations, Commun. Comput. Phys. 31 (2022) 370–397.
[18] M.L. Chiofalo, S. Succi, M.P. Tosi, Ground state of trapped interacting Bose-Einstein condensates by an explicit imaginary-time algorithm, Phys. Rev. E 62 (2000) 7438–7444.
[19] G. Cybenko, Approximation by superpositions of a sigmoidal function, Math. Control Signals Syst. 2 (1989) 303–314.
[20] I. Danaila, 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, B. Protas, Computation of ground states of the Gross-Pitaevskii functional via Riemannian optimization, SIAM J. Sci. Comput. 39 (2017) B1102–B1129.
[22] Y. Dauphin, R. Pascanu, C. Gulcehre, K. Cho, S. Ganguli, Y. Bengio, Identifying and attacking the saddle point problem in high-dimensional non-convex optimization, Adv. Neural Inf. Process. Syst. 27 (2014) 2933–2941.
[23] W. E, Machine learning and computational mathematics, Commun. Comput. Phys. 28 (2020) 1639–1670.
[24] W. E, J. Han, A. Jentzen, Deep learning-based numerical methods for high-dimensional parabolic partial differential equations and backward stochastic differential equations, Commun. Math. Stat. 5 (2017) 349–380.
[25] W. E, Q. Wang, Exponential convergence of the deep neural network approximation for analytic functions, Sci. China Math. 61 (2018) 1733–1740.
[26] W. E, B. Yu, The deep Ritz method: a deep learning-based numerical algorithm for solving variational problems, Commun. Math. Stat. 6 (2018) 1–12.
[27] L. Erdős, B. Schlein, H.T. Yau, Derivation of the Gross-Pitaevskii equation for the dynamics of Bose-Einstein condensate, Ann. Math. 172 (2010) 291–370.
[28] X. Glorot, Y. Bengio, Understanding the difficulty of training deep feedforward neural networks, in: Proceedings of the 13th International Conference on Artificial Intelligence and Statistics, vol. 9, 2010, pp. 249–256.
[29] J. Han, A. Jentzen, W. E, Solving high-dimensional partial differential equations using deep learning, Proc. Natl. Acad. Sci. 115 (2018) 8505–8510.
[30] G.E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, R.R. Salakhutdinov, Improving neural networks by preventing co-adaptation of feature detectors, CoRR, arXiv:1207.0580, 2012.
[31] S. Hon, Y. Sean, H. Yang, Simultaneous neural network approximation for smooth functions, Neural Netw. 154 (2021) 152–164.
[32] K. Hornik, Approximation capabilities of multilayer feedforward networks, Neural Netw. 4 (1991) 251–257.
[33] K. Hornik, Some new results on neural network approximation, Neural Netw. 6 (1993) 1069–1072.
[34] K. Hornik, S. Maxwell, W. Halbert, Universal approximation of an unknown mapping and its derivatives using multilayer feedforward networks, Neural Netw. 3 (1990) 551–560.
[35] K. Hornik, M. Stinchcombe, H. White, P. Auer, Degree of approximation results for feedforward networks approximating unknown mappings and their derivatives, Neural Comput. 6 (1994) 1262–1275.
[36] D.P. Kingma, J. Ba, Adam: a Method for Stochastic Optimization, ICLR, 2015, p. 13.
[37] A. Krizhevsky, I. Sutskever, G.E. Hinton, Imagenet classification with deep convolutional neural networks, Commun. ACM 60 (2017) 84–90.
[38] Y. LeCun, Y. Bengio, G. Hinton, Deep learning, Nature 521 (2015) 436–444.
[39] W. Liu, Y. Cai, Normalized gradient flow with Lagrange multiplier for computing ground states of Bose-Einstein condensates, SIAM J. Sci. Comput. 43 (2021) B219–B242.
[40] L. Lu, P. Jin, G. Pang, Z. Zhang, G.E. Karniadakis, Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators, Nat. Mach. Intell. 3 (2021) 218–229.
[41] E.H. Lieb, M. Loss, Analysis, Bull. Am. Math. Soc. (2001).
[42] M. Longo, S. Mishra, T.K. Rusch, Ch. Schwab, Higher-order quasi-Monte Carlo training of deep neural networks, SIAM J. Sci. Comput. 43 (2021) A3938–A3966.
[43] E.H. Lieb, R. Seiringer, J.P. Solovej, J. Yngvason, in: The Mathematics of the Bose Gas and Its Condensation, Birkhäuser Verlag, 2005.
[44] J. Lu, Z. Shen, H. Yang, S. Zhang, Deep network approximation for smooth functions, SIAM J. Math. Anal. 53 (2021) 5465–5506.
[45] Z. Li, H. Zheng, N. Kovachki, D. Jin, H. Chen, B. Liu, K. Azizzadenesheli, A. Anandkumar, Physics-informed neural operator for learning partial differential equations, ACM/IMS J. Data Sci. 3 (2024) 1–27.
[46] X. Liang, H. Zhang, S. Liu, Y. Li, Y. Zhang, Generation of Bose-Einstein condensates' ground state through machine learning, Sci. Rep. 8 (2018) 16337.
[47] O. Morsch, M. Oberthaler, Dynamics of Bose-Einstein condensates in optical lattices, Rev. Mod. Phys. 78 (2006) 179–215.
[48] L.P. Pitaevskii, S. Stringari, Bose-Einstein Condensation, Clarendon Press, Oxford, 2003.
[49] 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.
[50] H. Robbins, S. Monro, A stochastic approximation method, Ann. Stat. 22 (1951) 400–407.
[51] U. Shaham, A. Cloninger, R.R. Coifman, Provable approximation properties for deep neural networks, Appl. Comput. Harmon. Anal. 44 (2018) 537–557.
[52] J. Schmidhuber, Deep learning in neural networks: an overview, Neural Netw. 61 (2015) 85–117.
[53] J. Sirignano, K. Spiliopoulos, DGM: a deep learning algorithm for solving partial differential equations, J. Comput. Phys. 375 (2018) 1339–1364.
[54] Z. Shen, H. Yang, S. Zhang, Deep network approximation characterized by number of neurons, Commun. Comput. Phys. 28 (2020) 1768–1811.
[55] Z. Shen, H. Yang, S. Zhang, Optimal approximation rate of ReLU networks in terms of width and depth, J. Math. Pures Appl. 157 (2022) 101–135.
[56] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, in: 3rd IAPR ACPR, 2015, pp. 730–734.
[57] J. Williams, R. Walser, J. Cooper, E. Cornell, M. Holland, Nonlinear Josephson-type oscillations of a driven two-component Bose-Einstein condensate, Phys. Rev. A 59 (1999) R31–R34.
[58] Y. Zang, G. Bao, X. Ye, H. Zhou, Weak adversarial networks for high-dimensional partial differential equations, J. Comput. Phys. 411 (2020) 109409.
[59] S. Zeng, Z. Zhang, Q. Zou, Adaptive deep neural networks methods for high-dimensional partial differential equations, J. Comput. Phys. 463 (2022) 111232.