entropy

MDPI

Article

Deep Neural Network Model for Approximating Eigenmodes Localized by a Confining Potential

Luka Grubišić [1,*], Marko Hajba [2] and Domagoj Lacmanović [1]

[1] Department of Mathematics, Faculty of Science, University of Zagreb, 10000 Zagreb, Croatia; domagoj.lacmanovic@math.hr
[2] Department of ICT, Virovitica College, 33000 Virovitica, Croatia; marko.hajba@vsmti.hr
* Correspondence: luka.grubisic@math.hr

Abstract: We study eigenmode localization for a class of elliptic reaction-diffusion operators. As the prototype model problem we use a family of Schrödinger Hamiltonians parametrized by random potentials and study the associated effective confining potential. This problem is posed in the finite domain and we compute localized bounded states at the lower end of the spectrum. We present several deep network architectures that predict the localization of bounded states from a sample of a potential. For tackling higher dimensional problems, we consider a class of physics-informed deep dense networks. In particular, we focus on the interpretability of the proposed approaches. Deep network is used as a general reduced order model that describes the nonlinear connection between the potential and the ground state. The performance of the surrogate reduced model is controlled by an error estimator and the model is updated if necessary. Finally, we present a host of experiments to measure the accuracy and performance of the proposed algorithm.

Keywords: Anderson localization; deep neural networks; residual error estimates; physics informed neural networks

Citation: Grubišić, L.; Hajba, M.; Lacmanović, D. Deep Neural Network Model for Approximating Eigenmodes Localized by a Confining Potential. *Entropy* **2021**, *23*, 95. https://doi.org/10.3390/e23010095

Received: 29 November 2020
Accepted: 8 January 2021
Published: 11 January 2021

1. Introduction

In this paper we study features of the spectral problem for the family of elliptic operators of the reaction-diffusion type posed in the finite domain $\Omega = [-B, B]^n$, $B > 0$. These operators are also known as Schrödinger operators or Schrödinger Hamiltonians and they are defined by the differential expression of the form:

$$H_\omega u = -\triangle u + V_\omega u \, .$$

Here $-\triangle$ is the distributional realisation of the Laplace operator and V_ω is the multiplication operator with the real function V_ω. The parameter ω describes a random perturbation of a given potential. The associated spectral problem is to find an eigenvalue ε and an eigenmode $u \neq 0$ such that u verifies:

$$H_\omega u = \varepsilon u \tag{1}$$

and a set of boundary conditions. We will consider boundary conditions that lead to the realization of the expression H as a self-adjoint operator in a Hilbert space. In particular, we will consider functions $u \in H^1_0(\Omega)$ with Dirichlet boundary conditions and $u \in H^1_\pi(\Omega)$, where $H^1_\pi(\Omega)$ is the first-order Sobolev space of functions (square integrable functions whose gradient is also square integrable) that satisfy the periodic boundary conditions [1,2]. In what follows we will use $\|\cdot\|$ to denote the $L^2(\Omega)$ norm of a square integrable function.

We will consider—as an academic prototype—short-range confining electrostatic potentials such as those considered in [3] (see also [1,2]) and a more challenging class of confining potentials related to the effect of Anderson localization [4]. By the effect of localization we mean that we search for eigenvalues ε such that u, $\|u\| = 1$, is essentially

zero in the large part of the domain Ω. The Anderson model describes quantum states of an electron in a disordered alloy that can transition from metallic to insulating behavior. Loosely stated, we aim to model the connection $\mathcal{E} : V_\omega \mapsto u_\omega$, where u_ω, $u_\omega > 0$ is the eigenmode of the lowermost ε in (1). Such eigenmodes are called the ground states, and $\varepsilon_0 = \varepsilon$ is called the ground state energy. Note that for the elliptic reaction-diffusion operators H_ω defined in $L^2(\Omega)$ and with the potential V_ω, which is bounded, nonnegative and positive on a set of positive measures, there exists—by an application of the Krein–Rutman Theorem—the unique ground state $u_0 > 0$, which verifies (1) and so the mapping $\mathcal{E}$ is well defined; see [5,6].

We emphasize that we use the following regularization approach from [6] to deal with rough potentials. Namely for V_ω, which is bounded, nonnegative and positive on a set of positive measures there exists $u_\omega \in C^1(\Omega)$, $u_\omega > 0$ such that $H_\omega u_\omega = \mathbf{1}$. It can be shown by direct calculation that the operator A_ω defined in $H^1_\pi(\Omega)$ by the differential expression:

$$A_\omega \phi = -\frac{1}{u_\omega^2}\mathrm{div}(u_\omega^2 \mathrm{grad}\phi) + \frac{1}{u_\omega}\phi$$

has the same eigenvalues as the operator H_ω. Furthermore, ψ is an eigenmode of H_ω if and only if $\phi = \psi/u_\omega$ is an eigenmode of A_ω. Based on this equivalence, we call the function $W = 1/u_\omega$ the effective confining potential defined by V_ω. For more details see [5,7].

One possibility of obtaining data-sparse representations of the solutions of elliptic problems is through the use of tensor networks also known as tensor train decompositions or matrix product states [8,9]. We choose a more direct approach known as Variational Physical Informed Neural Network (VPINN) [10–12]. Realizations of these dense network architectures are trained to solve the variational formulation of the problem. This approach to training neural networks is a part of the unsupervised learning paradigm. It is a mesh-less approach that is capable of solving variational (physical) problems, by minimizing the loss functional, which combines the variational energy of the system together with penalty terms implementing further physical or normalization constraints.

The main physical constraint for the ground state is the positivity constraint. To deal with excited states we need to implement further symmetry constraints in the variational space. We opt for a different approach, also based on positivity constraints and a-posteriori error estimation. We use the neural network to approximate the solution of the source problem:

$$-\triangle u + Vu = \mathbf{1}$$

with associated boundary conditions. The solution u is called the landscape function, and in this case we are interested in the mapping $\mathcal{L} : V \mapsto u$. The landscape function is a positive function in $C^1(\Omega)$ and its reciprocal $W = 1/u$ is called the effective potential. The effective potential provides a mechanism that incurs localization on bounded states. To localize the excited states we use the approach of [5,7]. Let $W_{\min,i}$ be the i-th lowermost minimum of the effective potential W. It was observed in [5] that the following heuristic formula:

$$\tilde{\epsilon}_{i-1} = \left(1 + \frac{n}{4}\right) W_{\min,i} \tag{2}$$

yields good approximations to the energies of excited states. Note that this relationship, given its simplicity, is also something we might reasonably hope to learn algorithmically from a sample of landscape functions. This was stated as a motivation to utilize neural networks in the study of the eigenvalue problem for Schrödinger operators [3].

For an eigenmode ψ of H with the eigenvalue ε we have the estimate: $\psi(x) \leq \varepsilon u \|\psi\|_{L^\infty(\Omega)}$, $x \in \Omega$. This estimate can be obtained (see [13]) using the Feynman–Kac formula for the representation of the bounded state as an expectation of an integral of the Brownian motion. It was further argued in [13] that an eigenmode ψ with energy $\varepsilon := \varepsilon(\psi)$ can only localize in the region:

$$\{x \,:\, \varepsilon\, u(x) \geq 1\}\,. \tag{3}$$

Subsequently, as a combination of (2) and (3) we get both information on the excited state energy and information on the location of the excited state's support.

The Motivation and the Contribution of this Paper

The use of neural networks as data-sparse representations of complex, high dimensional nonlinear mappings is an emerging trend in many disciplines. In particular, it has been used to tackle many body Schrödinger equations [14,15], the Black–Scholes equation, the Hamilton–Jacobi–Bellman equation, and the Allen–Cahn equation [10,11,16–20].

In general, all of the above problems can be reduced to computing an approximation of the function $u : \Omega \subset \mathbb{R}^n \to \mathbb{R}$. This approximation is constructed by optimizing (training) the parameters of the family of test functions (we chose the family of all realizations of a given neural network architecture) so that the value of the appropriate energy functional (for the chosen model) is minimal. The main challenge in such an approach is to assess the approximation accuracy and to ensure that the computed realization of the neural network satisfies further physical constraints, such as symmetry constraint or the boundary conditions.

Further physical, but also numerical, constraints can be built into the optimization model in several ways. The most scalable and flexible way is to use penalization [10,11,20,21]. A alternative more subtle, and more accurate way is to introduce the constraints directly into the family of test functions as it is done in the architecture of the PauliNet from [14] (see also [22,23]), or to construct a family of test functions using an ansatz that combines several components of the solution, which are themselves realizations of neural networks [12,24].

In this study we focus on the potentials for which the Hamiltonian satisfies the Krein–Rutman theorem (the scaled ground state is the unique positive and smooth function). Examples of such potentials are the effective potentials associated with the Anderson localization. Since this is a more restrictive class of potentials than those considered in [14], we opt for a direct approach. Our contribution is the introduction of the residual error estimator into the Deep Ritz Algorithm from [20]. This in turn allows us to use Temple–Kato [25,26] or similar inequalities [27,28] to ascertain if the ground state generated by the neural network is a certified small perturbation of a physical eigenstate. For activation functions that are smooth enough we can calculate the strong form of the eigenvalue residual and then compute its norm using a quadrature or quasi-Monte Carlo integration. Using the residual estimator we stop the optimization (training process) when the eigenvalue residual is small enough (satisfies the preset tolerance) and/or the convergence criterion for the optimization algorithm is met (Adam optimizer). The use of ansatz functions based on neural networks, such as [24], will undoubtedly be a method of choice for 2D or 3D problems. However, this method depends on an accurate representation of the boundary of the domain Ω and thus faces challenges in scaling to higher dimensions.

The treatment of physical symmetries becomes critical when approximating excited states. For dealing with this task, we reformulate the problem as an inverse problem based on the solution of the source problem $Hu = \mathbf{1}$. The main constraint that the solution must satisfy is again positivity, and we construct an error estimator to certify the quality of the solution.

The network architectures used so far are dense network architectures. Inspired by [3], we study a parameter-dependent family of potentials and present a fully convolutional encoder–decoder neural network as a reduced order (surrogate) model for this family of partial differential equations and the mapping $\mathcal{L} : V \mapsto u$. We formulate a new certified surrogate modeling approach based on neural networks that is inspired by the work on certified surrogate modeling from [29] and the U-net architecture from [30]. We use the residual error estimator from the first part of the paper as a criterion for the surrogate (encoder–decoder) model update. For further details see Section 3.5.

Let us summarize the three classes of exemplar problems studied in this paper. First, we study the eigenvalue problem for approximating the unique positive normalized ground

state u_0. We aim to construct certified, robust and scalable—with respect to the increase in the dimension of the problem—approximation methods. Second, to approximate the eigenvalues higher in the spectrum we study the landscape function. The landscape function is obtained as a solution to the source problem $Hu = \mathbf{1}$. It is again a positive smooth function and positivity is the only physical constraint needed to study the localization phenomena for the associated eigenstates. Further, we use simple residual control to ensure that the computed solution is a small perturbation of the true landscape function. As the third class of problems, we present the data-based surrogate model of the map connecting a class of potentials to the associated landscape functions. Here we are concerned with the use of convolutional networks as a data-sparse reduced order model in the context of certified surrogate modeling of this mapping. In particular, we are interested in the possibility of updating the surrogate model based on the residual error estimator.

2. Theoretical Background

In order to be precise and explicit, we will present the theoretical foundations on a somewhat restricted set of neural network architectures. The network architectures that will be used in practical computations are presented in Appendix B. The change of the family of the realizations of neural networks over which the optimization is carried out does not change the presentation of the algorithms in any practical way. Our main contribution is in the introduction of the error control in the Deep Ritz algorithm from [20]. We will now summarize the basic definitions from [31], which are necessary to interpret the numerical experiments.

Definition 1. *Given $n_1, n_2, L \in \mathbb{N}$, a neural network θ of depth D with the input dimension n_1 and the output dimension n_2 is the D-tuple $\theta = \big((A_l, b_l) \,:\, l = 1, \cdots, D\big)$ where:*

$$(A_l, b_l) \in \mathbb{R}^{N_l \times N_{l-1}} \times \mathbb{R}^{N_l}, \;\; l = 1, \cdots, D.$$

By the convention $n_1 = N_0$ and $n_2 = N_D$. In the case in which $D = 2$ we call the network shallow, and otherwise the network is called deep. The vector $\vec{N} = \begin{pmatrix} N_0 & \cdots & N_D \end{pmatrix}$ is called the network architecture of the neural network θ.

We will use $D(\theta)$ to denote the depth of the given neural network θ. In the case in which the structure of matrices A_l, $l = 1, \cdots, D$ is not further restricted, we say that the network is dense. In the case in which a sparsity pattern is assumed we have several subclasses of neural networks. For exemple, if the matrices A_l, $l = 1, \cdots, D$ have a structure of a Toeplitz matrix $(A_l)_{ij} = (w^l_{i-j})_{ij}$—here w^l, $l = 1, \cdots, D$ are parameter vectors defining a Toeplitz matrix [32]—we talk about convolutional neural networks.

Let $\rho : \mathbb{R} \to \mathbb{R}$ be a function that is not a polynomial. By $\boldsymbol{\rho}$ we denote the function $\boldsymbol{\rho} = \otimes^n_{l=1}\rho : \mathbb{R}^n \to \mathbb{R}^n$. We will now define a realization of the neural network θ with respect to the function ρ.

Definition 2. *A function $R_{\theta,\rho} : \mathbb{R}^n \to \mathbb{R}^m$ is defined by the formula:*

$$R_{\theta,\rho}(x) = T_L(\boldsymbol{\rho}(T_{D-1}(\boldsymbol{\rho}(T_{D-2}(\cdots \boldsymbol{\rho}(T_1(x)))))))\,.$$

where $T_l(x) = A_l x + b_l$, $l = 1, \cdots, D$ is called the realization of the neural network θ with respect to the activation function ρ.

Among various activation functions we single out the rectified linear unit (ReLU) function $\rho_{LU} = \max\{0, x\}$ and the sigmoid function $\rho_S(x) = 1/(1 - \exp(-x))$. The set of all ReLU realizations of a neural network θ has a special structure. We call a function $f : \mathbb{R}^{n_1} \to \mathbb{R}^{n_2}$ piece-wise linear if there is a finite set of pairwise disjoint, closed polyhedra whose union is $\mathbb{R}^{n_1}$ such that a restriction of f onto a chosen polyhedron is an affine function. It has been shown in [33] that any piece-wise linear function can be represented

by a ReLU neural network and that any ReLU realization of a neural network is piece-wise linear. This observation is key to linking the approximation theory for neural networks with the standard Sobolev space regularity theory for partial differential equations.

Let us now fix some further notation. Let m be the number of the degrees of freedom of the space of piece-wise linear functions associated to the fixed polyhedral tessellation of Ω. We use V_m to denote the set of all piece-wise linear functions on this tessellation. We also use the notations P_1, P_2 and P_3 to denote the space of piece-wise linear, quadratic and cubic functions, respectively. The corresponding interpolation operators (for continuous arguments) are denoted respectively by I_{P_1}, I_{P_2} and I_{P_3}.

We will now briefly review a-posteriori error estimates that are used in this work. Let us note the following convention. We use ε_0 and u_0 to denote the ground state energy and the normalized positive ground state. We use ε_1 to denote the energy of a first excited state and we note that the notation is generalized for higher excited states in an obvious way. We denote the Rayleigh quotient of the operator H for the state ψ by $\varepsilon = \varepsilon(\psi) := (\psi, H\psi)/(\psi, \psi)$. The standard Kato–Temple estimate from [26] can be written in a dimension-free form, also known as the relative form:

$$\frac{|\varepsilon - \varepsilon_0|}{|\varepsilon|} \leq \frac{|\varepsilon|}{|\varepsilon_1 - \varepsilon|} \frac{\|H\psi - \varepsilon\psi\|^2}{|\varepsilon|^2 \|\psi\|^2}. \tag{4}$$

The quantity $|\varepsilon|/|\varepsilon_1 - \varepsilon|$ is a measure of the so-called relative spectral gap [34,35] and it measures the distance to the unwanted part of the spectrum. It can be estimated by symmetry considerations or other a-priori information. In fact, a more careful analysis from [35] shows that the scaled residual is an asymptotically exact estimate of the relative error and so we will heuristically drop the measure of the gap even in the preasymptotic regime. A consequence of the Davis–Kahan theorem [36] is that the residual also estimates the eigenvector error:

$$\frac{\|\psi - u_0\|}{\|\psi\|} \leq c \frac{|\varepsilon|}{|\varepsilon_1 - \varepsilon|} \frac{\|H\psi - \varepsilon\psi\|}{|\varepsilon| \|\psi\|}. \tag{5}$$

Subsequently, the error estimator $\Delta_\varepsilon^2 = \|H\psi - \varepsilon\psi\|^2/(\varepsilon^2\|\psi\|^2)$ is a good stopping criterion for a certified approximation of an eigenmode.

For the source problem $Hu = \mathbf{1}$, $u \in H_0^1(\Omega)$ we note the following relationship:

$$\frac{\|u - \underline{u}\|_{L^2}}{\|\underline{u}\|_{L^2}} \leq \frac{\|\mathbf{1}\|_{L^2}}{\varepsilon_0} \frac{\|H\underline{u} - \mathbf{1}\|_{L^2}}{\|\underline{u}\|_{L^2} \|\mathbf{1}\|_{L^2}},$$

between the residual and the relative L^2 error. Subsequently we use $\tau = \|H\underline{u} - \mathbf{1}\|_{L^2}/(\|\underline{u}\|_{L^2}\|\mathbf{1}\|_{L^2})$ as an error estimator for the source problem.

Algorithms

We will now present the modifications of algorithms that we used to study the localization phenomena. We modified the Deep Ritz algorithm from [20] with the introduction of the a-posteriori (residual) error estimator. We call our variant the Certified Deep Ritz Algorithm. It is motivated by the work on certified reduced-order modeling in [29]. In order to be able to formulate strong residuals, we chose smooth activation functions ρ, so that $R_{\theta,\rho}$ can be used to form the strong residual.

The performance of the stochastic gradient descent, when applied to the loss function, can be highly sensitive to the choice of the learning rate. Furthermore, it can also suffer from oscillation effects introduced by the choice of the sampling method in the numerical integration routines. For this reason we have opted to use the Adam optimizer from [37], which determines the learning rate by adaptively using information from higher-order moments.

To enforce the positivity constraint, we composed a realization of the neural network with the function Ξ, $\Xi > 0$. We call the function Ξ a positivity mask and it must be chosen appropriately for the governing boundary conditions. As the positivity mask for

Schrödinger Hamiltonians we either chose a smooth nonnegative function Ξ, which decays to zero as $|x| \to \infty$, or set the positivity mask as the identity.

In Algorithm 1 the parameter $\beta > 0$ is the penalty parameter used to enforce the boundary conditions and the parameter $\eta > 0$ is used to normalize the eigenmode approximation. We solve the integral using a Gauss quadrature rule in 1D and for higher dimensional problems we use quasi-Monte Carlo integration from [38,39] or a sparse grid quadrature [40] to approximate the integrals in the loss function as well as for the final (more accurate) evaluation of the energy functional. Alternatively in 2D, we sometimes choose to compute the integrals by projecting the realization of a neural network into a finite element space and then use the finite element quadrature to compute the integral. According to the authors of [11,41], this is an appropriate approach for problems of moderate dimensions ($\Omega \subset \mathbb{R}^n$, $n \leq 20$). For higher-dimensional problems, Monte Carlo integration is the only scalable approach recommended.

Algorithm 1: Certified Deep Ritz Algorithm.

Result: Approximation $\psi_{\theta,\rho}$ of the ground state
Choose the activation function ρ, the positivity mask Ξ, the penalty parameters β, η, the maximal number of iterations `maxit`, the network architecture $\vec{N}$, the integration method and the residual measure $\tilde{\Delta}$;
Set $\tilde{\Delta} = 1$ and chose the starting θ with the architecture $\vec{N}$.
while $\tilde{\Delta}$ *is not small enough and the number of iterations is strictly smaller than* `maxit` **do**
  Set $\psi_{\theta,\rho} = \Xi \circ R_{\theta,\rho}$ and compute $L(\psi_{\theta,\rho}, \beta, \eta)$ using numerical integration.
  Update θ using the Adam optimizer for the loss function $L(\psi_{\theta,\rho}, \beta, \eta)$.
  Compute $\tilde{\Delta}$ using numerical integration.
end
if `maxit` *reached* **then**
  Deepen $\vec{N}$ and start over.
end

To apply Algorithm 1 to an eigenvalue problem we choose:

$$\tilde{\Delta} := \Delta_\varepsilon^2 = \|H\psi_{\theta,\rho} - \varepsilon(\psi_{\theta,\rho})\psi_{\theta,\rho}\|^2 / (\varepsilon(\psi_{\theta,\rho})^2 \|\psi_{\theta,\rho}\|^2)$$

and set the normalization parameter $\eta > 0$ and $\beta > 0$ for the loss function:

$$L(u; \beta, \eta) = \frac{\int_\Omega |\nabla u|^2 + \int_\Omega V_\omega u^2}{\int_\Omega u} + \beta \|u\|^2_{L^2(\partial\Omega)} + \eta \left(\int_\Omega u^2 - 1 \right). \tag{6}$$

In the case of the source problem for the computation of the landscape function we set the normalization parameter $\eta = 0$ and $\beta > 0$ and define the loss function as:

$$L(u; \beta, \eta) = \frac{1}{2} \left(\int_\Omega |\nabla u|^2 + \int_\Omega V_\omega u^2 \right) - \int_\Omega u + \beta \|u\|^2_{L^2(\partial\Omega)}.$$

As the error indicator we take $\tilde{\Delta} = \tau = \|H\underline{u} - \mathbf{1}\|_{L^2} / (\|\underline{u}\|_{L^2} \|\mathbf{1}\|_{L^2})$, where $\underline{u} = \Xi \circ R_{\theta,\rho}$. Here we have chosen as an example the homogeneous Dirichlet boundary condition $u|_{\partial\Omega} = 0$. Other self-adjoint boundary conditions can equally be implemented by penalizing the boundary conditions residual at the boundary $\partial\Omega$. Note that computing derivatives of realizations of neural networks is efficiently implemented in many programming environments such as TensorFlow [42].

3. Results

In this section we present direct approximation methods for estimating the ground state u_0, the ground state energy ε_0 and the landscape function u. We use the Certified Deep Ritz Algorithm presented as Algorithm 1.

3.1. Direct Approximations of the Ground State in 1D

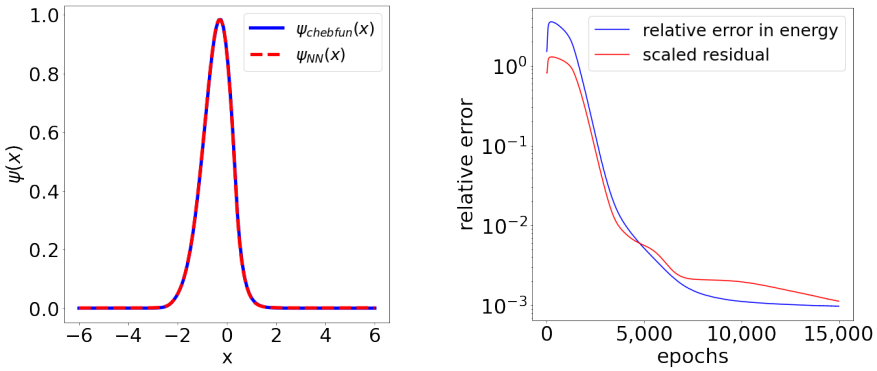
We now present the results of the application of Algorithm 1 on the 1D Schrödinger operator $H = -\partial_{xx} + V$. For the domain $\Omega = (-B, B)$, we choose the loss function (6) and set $\Xi(x) = \exp(-x^2/10)$ as the positivity mask in Algorithm 1.

To benchmark the accuracy of the VPINN approximations we have solved the problem to high relative accuracy using the Chebyshev spectral method as implemented in the package `chebfun` [43,44]. We emphasize that `chebfun` was not used during the training of the network in any way. To compute the residuals and the energy of the ground state we used a Gaussian quadrature where the deep network is evaluated at the sufficient number—for the given interval $(-B, B)$—of Gaussian points.

We constructed the potential V as a linear combination of the finite well and two inverted Gaussian bell functions:

$$V = -\alpha_1 \exp(-\|\cdot - c_1\|)^2/k_1^2) - \alpha_2 \exp(-\|\cdot - c_2\|)^2/k_2^2) - h\mathbf{1}_{\{x:\ |x-c|<t\}}.$$

We used $\mathbf{1}_{\{x:\ |x-c|<t\}}$ to denote the indicator function and chose $\alpha_i \in [8, 12]$, $c_i \in [-2.5, 2.5]$, $k_i \in [0.9, 2.6]$, $i = 1, 2$, $h \in [10, 15]$, $c \in [-2, 2]$, $t \in [0.5, 2.5]$. The neural network has 1,162 trainable parameters and we used the `DenseNet` VPINN architecture $\vec{N}_{\texttt{DenseNet}}(1, 4, 2, 10)$ with the activation function $\rho(x) = \exp(-x^2/10)$; see Figure A1. Figure 1 shows the solution and the error estimate during 15,000 epochs of a run of the Adam optimizer with the learning rate $\delta = 10^{-3}$ and a batch size of 1024. In this example we used 1024 quadrature points on an interval $(-6, 6)$ and the penalty parameters $\beta = 10^{-3}$ and $\eta = 20$.



(**a**) (**b**)

Figure 1. (**a**) Comparison of the ground state obtained in `chebfun` ($\psi_{chebfun}(x)$) and as the VPINN solution ($\psi_{NN}(x)$) with the architecture $\vec{N}_{\texttt{DenseNet}} = (n, k, l, m) = (1, 4, 2, 10)$; (**b**) Residual and Rayleigh quotient error estimate metrics during the training process.

One can observe robust, almost asymptotically exact, performance of the estimator:

$$\Delta_\varepsilon^2 = \|H\psi - \varepsilon\psi\|^2/(\varepsilon^2\|\psi\|^2)\,.$$

Let us also emphasize that Δ_ε measures the distance of the Rayleigh quotient (energy functional) to the nearest eigenvalue. For evaluation of the integrals in higher dimensions we refer the reader to Appendix B. Note that the final error for the approximation of the ground state energy was 0.1%, whereas the final relative L^2 error in the ground state was 0.9%. This is in line with the eigenmode error estimate (5).

3.2. Direct Approximations of the Ground State in Higher-Dimensional Spaces

We present VPINN approximation results for the ground state and the ground state energy of the Schrödinger equation with harmonic oscillator potential $V(x) = \|x\|^2$ using the dense network with the architecture depicted in Figure A1. We study the problem on the truncated domain $[-3,3]^n$, where n is the dimension of the space. Neural network architecture should be constructed with caution. There are multiple sources of instability when dealing with neural networks, e.g., exploding and vanishing gradients. We experimented with a variety of different activation functions: $\rho_S(x)$, $\rho_{LU}(x/10)^2$, $x\rho_S(x)$, $\exp(-x^2/10)$ etc. After training of the neural network using the quasi-Monte Carlo realization of the energy integrals to define the loss function we computed the approximate ground state energy using the approximation of the energy functional (Rayleigh quotient) using the Sobol sequences with 100,000 points. We also report on the results obtained using Smolyak grids of order 6 with the Gauss–Patterson rule. The results are presented in Table 1.

Table 1. Convergence rates for the ground state energy of the harmonic oscillator in relation to the dimension. QMC: quasi-Monte Carlo.

n	ε_0	M for the Loss Function	Adam Optimizer Epochs	M for the Smolyak Quadrature	Smolyak Relative Error %	Relative Error for QMC with $M = 10^5$ Points%
1	1	100	50,000	127	0.004	0.003
2	2	1000	20,000	769	1.416	1.226
3	3	5000	50,000	2815	1.110	1.608
6	6	50,000	80,000	40,193	-	1.40
9	9	50,000	50,000	242,815	230.366	5.816

The accuracy of the ground state energy approximations that were obtained using quasi-Monte Carlo integration are comparable with the accuracy reported in [20]. On the other hand, the results obtained by using Smolyak's points were unsatisfactory in dimensions higher than 3. This observation will be the subject of future research. It appears that the oscillation of the realizations of the neural network on the boundary of the computational domain together with the appearance of negative weights in sparse grid integral formulas contributed to the instability of the approach. We used the $\vec{N} = (n, 5, 2, 20)$ architectures for $n \leq 6$ and $\vec{N} = (9, 3, 1, 10)$ in 9D and the swish function $\rho(x) = x\rho_S(x)$ as the activation function. The positivity mask was chosen as the identity.

3.3. Approximations of the Landscape Function in 1D

We now present the result of the approximation method using the landscape function as a solution of the partial differential equation $Hu = \mathbf{1}$, $u \in H_0^1(\Omega)$. The potential $V \in C([-50, 50])$ is constructed as a random piece-wise linear function (see Figure 2). More to the point, let ξ_k, $k = -50, \cdots, 50$ be independently drawn numbers from the uniform distribution on the interval $[0, 4]$. We construct the potential V as the piece-wise linear interpolant of (k, ξ_k), $k = -50, \cdots, 50$. We again used `chebfun` for benchmarking. We set $\eta = 0$ and defined the loss function as:

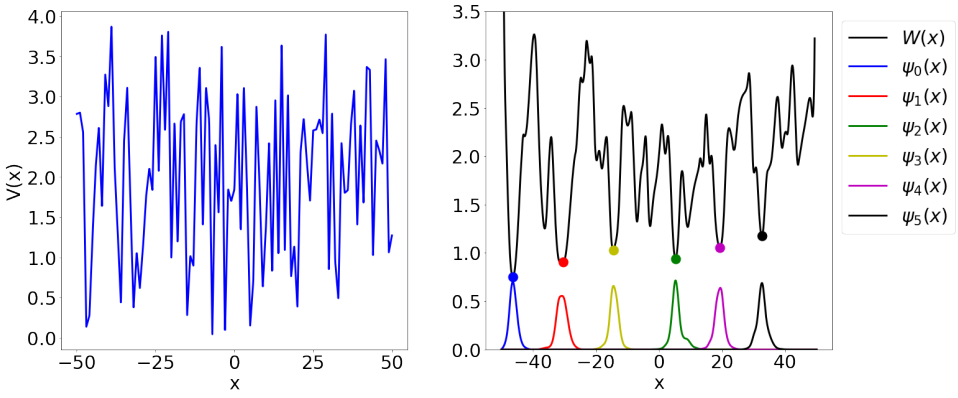
$$L(u; \beta, \eta) = \frac{1}{2}\left(\int_\Omega |\nabla u|^2 + \int_\Omega V_\omega u^2\right) - \int_\Omega u + \beta(u(-50)^2 + u(50)^2)\,. \tag{7}$$

Here we have used the architecture of the dense VPINN network; see Appendix C. We used $\beta = 500$ for the experiments. In Figure 2 we can see the six local minima of the effective potential $W = 1/u$ obtained from the neural network and the first six eigenstates computed by the chebfun. Note that the potential $W = 1/u$ is defined only in the interior of the domain $(-50, 50)$. In Table 2 we present the results of the benchmarking of the approximation formula (2) against highly accurate `chebfun` eigenvalue approximations.

Table 2. We tested the accuracy of the predictor $\tilde{\epsilon}_{i-1} = \left(1 + \frac{n}{4}\right) W_{min,i}$ for 16 lowermost eigenvalues. The `chebfun` solution was used to benchmark the error.

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Minimum values of W	0.747201	0.918677	0.918754	0.933014	1.028903	1.057663	1.174706	1.245278
chebfun eigenvalues	0.979730	1.071839	1.230230	1.282611	1.301724	1.485232	1.577349	1.588252
Relative error in %	4.6675	7.1379	6.6481	9.0708	1.1981	1.9850	6.9082	1.9930
	8	**9**	**10**	**11**	**12**	**13**	**14**	**15**
Minimum values of W	1.256498	1.273980	1.326926	1.613203	1.848415	1.868003	1.907063	1.931723
`chebfun` eigenvalues	1.625253	1.758768	1.780166	2.095899	2.161778	2.265704	2.270798	2.278380
Relative error in %	3.3614	9.4551	6.8257	3.7882	6.8805	3.05864	4.9776	5.9811

The neural network has 6402 trainable parameters and we used the VPINN architecture $\vec{N}_{\texttt{DenseNet}} = (1, 5, 2, 20)$ with the activation function $\rho_{LU}(x/10)^2$. The positvitiy mask was chosen as the identity. The network was trained using 50,000 epochs of the Adam optimizer with the learning rate $\delta = 10^{-3}$ and a batch size of 2048. In this example we also used 2048 quadrature points on an interval $(-50, 50)$.



(**a**) Random potential (**b**) The effective potential

Figure 2. The effective potential and its 6 local minima, which define localization of the first six eigenstates is shown on the right. Eigenstates $\psi_i, i = 0, 1, ..., 5$ were computed in chebfun.

3.4. Direct VPINN Approximation of the Landscape Function in 2D

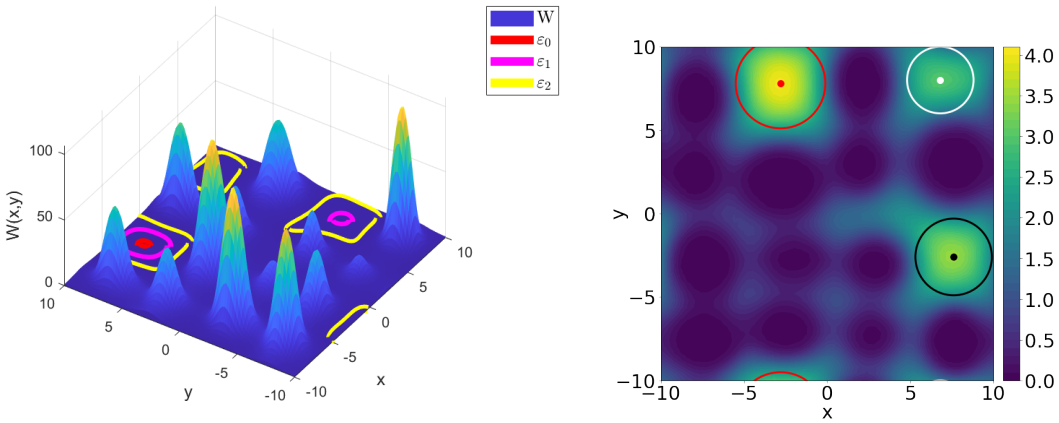
We now apply Algorithm 1 to the problem of approximating the landscape function in 2D. When presenting the examples we will report on the used network architecture $\vec{N}_{\texttt{DenseNet}} = (2, k, l, m)$ as well as indicate the number of trainable parameters for each of the architectures. The activation function used for all neural networks in this subsection is the sigmoid function ρ_S. We now set $l = 2$ and in the next table present a convergence study for the family of architectures $\vec{N}_{\texttt{DenseNet}} = (2, k, 2, m)$.

The convergence histories of relative L^2 and H^1 errors, measured with respect to the benchmark FEniCS solution, are shown in Table 3. We can observe that the errors drop at a favorable rate with an increase in k. On the other hand, an increase in m causes a much more pronounced increase in the number of trainable parameters (complexity of the network) but incurs, in comparison, only a moderate improvement of the accuracy level.

Table 3. A report on the convergence in k and m for the family of architectures $\vec{N}_{\texttt{DenseNet}} = (2, k, 2, m)$. We benchmark the error against the highly accurate P_3 FEniCS solution.

Parameters	k	m	Relative L^2 Error 100,000 Epoch	Relative H^1 Error 100,000 Epoch	Relative L^2 Error 200,000 Epoch	Relative H^1 Error 200,000 Epoch	Relative Error of the First Three Eigenvalues Respectively
803	4	8	2.5852%	5.6216%	2.0527%	4.9876%	0.1638%, 1.4479%, 1.1472%
1203	4	10	2.7487%	5.3611%	1.2354%	3.6960%	0.0839%, 2.3489%, 0.6341%
1753	5	10	1.9314%	4.2386%	1.0679%	3.3851%	0.5957%, 1.9264%, 0.3822%
2403	6	10	1.1745%	3.0548%	0.7998%	2.6994%	0.4539%, 1.7883%, 1.5112%
4403	4	20	1.9037%	3.6929%	0.7233%	2.5757%	0.3242%, 1.8831%, 1.2586%
9603	4	30	1.8217%	3.7451%	0.6689%	2.3609%	0.3639%, 2.0083%, 0.9685%
16,803	4	40	0.6372%	1.9704%	0.3920%	1.5497%	0.3269%, 1.8606%, 0.6983%
26,003	4	50	3.6993%	7.3510%	0.4207%	1.6748%	0.3127%, 1.5756%, 0.3559%

In Figure 3 we plot the effective potential W and the landscape function u.



(**a**) (**b**)

Figure 3. A surface plot of the effective potential $W = 1/u$ (**a**) and the landscape function u (**b**). In (**a**) we plot the boundaries of the sets $\{x : \varepsilon\, u(x) \geq 1\}$ that localize the eigenstates. In (**b**) we plot the circles of radius $1/\tilde{\varepsilon}_i$, for $\tilde{\varepsilon}_{i-1} = 3W_{\min,i}/2$, $i = 1, 2, 3$, centered at the i-th lowermost local minimum $W_{\min,i}$.

3.5. Encoder–Decoder Network as a Reduced-Order Model for a Family of Landscape Functions

We now study the use of the sparse, U-Net-inspired [30], network architecture as a surrogate model for the function $\mathcal{L} : V \mapsto u$. In Figure 3 we show the landscape function u with periodic boundary conditions for the potential V_ω constructed as a lattice superposition of sixteen Gaussian bell functions $G(x) = \alpha \exp(-|x_1 - c_1|^2/k_1^2 - |x_2 - c_2|^2/k_2^2)$. The centers $c = (c_1, c_2)$ were chosen randomly inside each block of the 4×4 uniform quadrilateral tessellation of Ω. The constants $\alpha, k_i, i = 1, 2$ were chosen randomly

and independently from intervals $[8, 128]$ and $[1, 1.5]$, respectively. To introduce local defects in the lattice, we have further randomly chosen three Gaussian bells and removed them from the potential. The choice of Gaussian bells to be removed was restricted, so that the boundary conditions were respected and that none of the erased bells were pairwise adjacent.

As the reduced order model for this family of problems, we have used the encoder–decoder fully convolutional neural network (FCNN) from Appendix C with 2,614,531 trainable parameters. This is a relatively small number of parameters in comparison with the typical convolutional neural network architectures with fully connected layers. The architecture of the neural network is shown in Figure A2.

To train the model we generated 98,400 potentials V_ω and then used FEniCS to compute the associated landscape functions u_ω, $-\triangle u_\omega + V_\omega u_\omega = \mathbf{1}$. The domain of the Hamiltonaian was $\Omega = [-10, 10]^2$, with the periodic boundary conditions. We used the uniform quadrilateral discretization with the step size $h = 20/50 = 0.4$ and P_2 elements to compute the training examples. To construct the reduced-order model, we projected (by interpolation) these P_2 functions onto the space of P_1 elements for the same mesh. After implementing the periodic boundary conditions we obtained exactly 2,500 free nodes for this space of P_1 functions.

We denote the values of the potential V in those nodes as the vector $\vec{V} \in \mathbb{R}^{2,500}$ and we tacitly identify the vector $\vec{V}$ with the function $I_{P_1} V$. Let $I^*_{P_1}$ be the extension operator from $\mathbb{R}^{2,500}$ to the space of continuous piece-wise linear functions. Then

$$\tilde{\mathcal{L}}(\vec{V}) := I_{P_1} \mathcal{L}(I^*_{P_1} I_{P_1} V)$$

defines the mapping $\tilde{\mathcal{L}} : \mathbb{R}^{2,500} \to \mathbb{R}^{2,500}$. We used the learning rate $\delta = 10^{-4}$ for the first 100 epochs of the Adam optimizer and the learning rate $\delta = 10^{-5}$ for a further 50 epochs. The activation function ρ_{LU} was used to promote sparsity and the MSE loss function was used for the training. The batch size for the Adam algorithm was 1024. We implemented the certified surrogate modeling approach by combining the evaluation of the neural network with the error estimator τ. In the case in which the residual measure τ for the function $\underline{u} = \mathcal{L}(I^*_{P_1} I_{P_1} \hat{V})$ is larger than the preset tolerance, we updated the surrogate model (neural network). To this end we solved in FEniCS the problem $-\triangle u + \hat{V} u = \mathbf{1}$ and used the standard update algorithm for the convolutional neural network and the new training example.

We evaluated the performance of the neural network reduced-order model on a set of 200 testing potentials that were not used in the training of the network, see Table 4. The benchmarking comparison against the FEniCS solution is presented in Figure 4.

Table 4. Validation of the encoder–decoder representation of the mapping $\mathcal{L} : V \mapsto u$ on a collection of test examples. Recall that the effective potential is defined as $W = 1/u$.

Average L^2 error	1.7545%
Maximal L^2 error	2.9769%, example: 58
Average H^1 error	9.2233%
Maximal H^1 error	12.6765%, example: 65
Mean relative error in $1/W_{\min,1}$	0.4887%
Maximal relative error in $1/W_{\min,1}$	2.1402%, example: 70
The worst ten relative errors in $1/W_{\min,1}$ (%)	2.1402, 1.5909, 1.5560, 1.4816, 1.4151, 1.4626, 1.3441, 1.3377, 1.3181, 1.3132

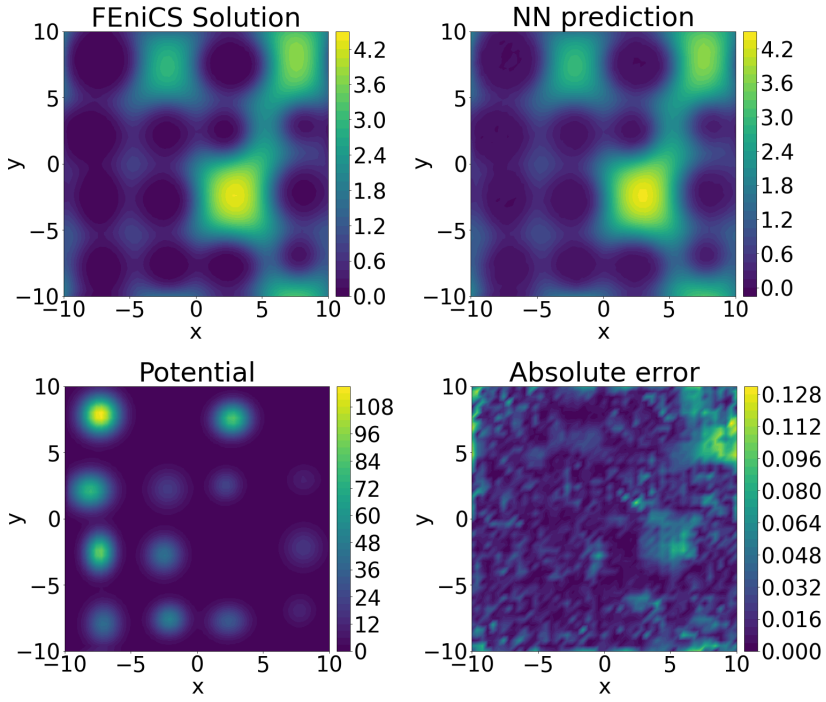


Figure 4. A benchmarking comparison of the encoder–decoder prediction of the landscape function against the FEniCS solution.

4. Discussion

According to the authors of [45], deep learning approaches to dealing with partial differential equations fall into the following three categories: (a) Rayleigh–Ritz approximations, (b) Galerkin approximations and (c) least squares approximations. We have considered a hybrid approach that combines robust stochastic optimization of overparametrized networks based on the Rayleigh–Ritz approach with standard residual based error estimates, which together yield a hybrid approximation method. We were particularly influenced by the review in [45] and the Deep Ritz algorithm as described in [20].

In the example from Table 3 we further computed a piece-wise cubic and piece-wise quadratic approximation of the landscape function using the standard finite element method and an approximation of the landscape function using a variant of the Deep Ritz method. We measured the error of the P_2 and the VPINN approximation against the P_3 benchmark solution. The relative L^2 error of the piece-wise quadratic approximation was computed to be 0.36%, whereas the relative error of the VPINN approximation with 1203 free parameters was computed to be 1.21%. Note, however, that the piece-wise quadratic approximation required the training of 10,000 parameters. This indicates that the neural network achieves a considerable data compression when compared with a piece-wise polynomial approximation. The situation is even more interesting in 1D. There we compared the ground state approximation by the Chebyshev series, as implemented in `chebfun`. We needed 149 terms in the Chebyshev expansion to reach the order of machine precision. On the other hand, the Adam optimizer was able to find a realization of the dense neural network with VPINN architecture $\vec{N}_{\texttt{DenseNet}} = (1, 2, 2, 2)$ from Appendix A. This architecture only has 30 trainable parameters and it achieves a relative distance (in the L^2 sense) of 1.05% to the benchmark `chebfun` solution. The relative error in the ground state energy is only 0.1%, see Figure 5.

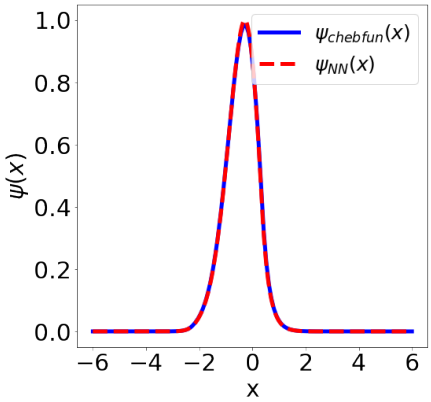


Figure 5. Comparing the Chebyshev series expansion with 149 terms and a VPINN solution with the architecture $\vec{N}_{\texttt{DenseNet}} = (1, 2, 2, 2)$ and 30 trainable parameters.

The integrals needed to approximate the energy functional were computed using Gaussian quadrature rules. Unfortunately, this approach does not yield stable methods in higher-dimensional problems. The reason is in the fact that sparse grid quadratures (e.g., [40]) also have nodes with negative weights and this was observed to cause severe numerical instability. An approach based on the quasi-Monte Carlo integration, which utilizes low-discrepancy sequences of integration nodes and has only positive weights (see [39]) yielded an efficient and stable method in higher-dimensional situations. Furthermore, since realizations of neural networks are frequently functions with many local extrema, computing their integrals needs to be handled with care. This is particularly relevant when enforcing discretizations of physical or normalization constraints by penalization. We point out that the scalability of sparse-grid integration schemes in this respect was not satisfactory.

Another promising technique for obtaining data-sparse compressed approximation of the solutions of partial differential equations is based on the concept of tensor networks, also known as matrix product states or tensor train decompositions [9]. This approach has been successfully converted into numerical approximation algorithms such as the quantized tensor train decompositions [8,46]. The scaling robust performance of this approach has been demonstrated on a class of multi-scale model problems in [46]. However, the numerical methods still have to be tailor-made for the chosen problems. On the other hand, there are many freely available robust and highly developed libraries for working with deep neural networks. This is the reason for our choice of the discretization method.

5. Conclusions

We have presented two types of neural network architectures. First, a dense deep network of the DenseNet type was used as a compressed approximation of the ground state and the landscape function of the problems under consideration. Remarkably, it achieved high accuracy and a good compression rate even when empirically compared with a Chebyshev expansion in 1D. Even though we managed to tackle problems in $\mathbb{R}^n$, this concept struggled to yield scalable numerical methods. We then took another approach and considered a problem of approximating a mapping $\mathcal{L} : \mathbb{R}^{2500} \to \mathbb{R}^{2500}$, which connects a mesh sample of a potential with the associated landscape function. A fully convolutional neural network architecture with the ReLU activation function, to further promote sparsity, turned out to be expressive enough to deal with this family of problems to a satisfactory level of accuracy (empirically measured on the test set). We have also seen that a hybrid approach—one that combines the expressivity of the set of neural network realizations with the standard error indicators—has a potential to lead to robust approximation methods. We have observed that it is particularly challenging to turn physical constraints—which are continuous—into their discrete realizations, which can be used to filter out, e.g., by judiciously applied penalization, the nonphysical neural network realizations from the

set of all realizations of a given architecture. How to turn this into a robust mesh-less and scalable method for dealing with equations of mathematical physics will be a topic of further research.

Author Contributions: Conceptualization, L.G.; methodology, L.G., M.H., D.L.; software, L.G, M.H. and D.L.; validation, L.G., M.H. and D.L.; formal analysis, L.G.; investigation, M.H. and D.L.; resources, L.G.; data curation, M.H.; writing—original draft preparation, L.G.; writing—review and editing, L.G., M.H. and D.L.; visualization, M.H. and D.L.; supervision, L.G.; project administration, L.G.; funding acquisition, L.G. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Hrvatska Zaklada za Znanost (Croatian Science Foundation) under the grant IP-2019-04-6268—Randomized low rank algorithms and applications to parameter dependent problems.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

Sample Availability: Codes are available from the GitHub repository https://github.com/markohajba/NN-Schrodinger. All communication regarding software should be directed to marko.hajba@vsmti.hr.

Abbreviations

The following abbreviations are used in this manuscript:

PDE	partial differential equation
ReLU	rectified linear unit
FEM	finite element method
DOF	degrees of freedom
VPINN	Variational Physics Informed Neural Networks
FCNN	fully convolutional neural network

Appendix A. Implementation Details

In the implementation of the discussed methods we have used the following tools in the Python programming environment: TensorFlow 2 [42], Keras [47], chaospy [38] and FEniCS [48]. Implementations and additional materials are available at GitHub repository https://github.com/markohajba/NN-Schrodinger.

Appendix B. Estimating Residuals

The Rayleigh quotient of the operator H for the mode ψ is computed by evaluating the integral:

$$\varepsilon(\psi) = \frac{\int_\Omega \nabla\psi \cdot \nabla\psi + V\psi^2}{\int_\Omega \psi^2} .$$

The eigenvalue residual for the mode ψ is the functional:

$$\phi \mapsto (H\psi - \varepsilon(\psi)\psi, \phi) = \int_\Omega (\nabla\psi \cdot \nabla\phi + (V - \varepsilon(\psi))\psi\phi) .$$

We measure the norm of the residual by approximating the supremum:

$$\sup_{\phi \in \mathcal{S} \subset H^1(\Omega)} \frac{|(H\psi - \varepsilon(\psi)\psi, \phi)|}{\|\phi\|}$$

over a judiciously chosen set $\mathcal{S}$. This residual can also be written as a functional and approximated by solving the optimization problem over $\mathcal{S}$. We leave out the details.

In the original Deep Ritz Algorithm from [20] the authors have used Monte Carlo integration to compute the duality products. This amounts to choosing a random sample $\chi_i \in \Omega$, $i = 1, \cdots, M$ and then using:

$$\int_\Omega \|\nabla\psi\|^2 + V\psi^2 \approx \frac{1}{M}\sum_{i=1}^{M} (\|\nabla\psi(\chi_i)\|^2 + V\psi(\chi_i)^2) \,.$$

By contrast, in $\mathbb{R}^n$, we used the Smolyak quadrature [38,40] and low-discrepancy sequences for quasi-Monte Carlo integration routines [39]. For a given order of the quadrature there exist weights $\alpha_i \in \mathbb{R}$ and nodes $\xi_i \in \Omega$, $i = 1, \cdots, M$ such that:

$$\int_\Omega \|\nabla\psi\|^2 + V\psi^2 \approx \sum_{i=1}^{M} \alpha_i(\|\nabla\psi(\xi_i)\|^2 + V\psi(\xi_i)^2) \,.$$

Unlike in the Monte Carlo approach, these nodes are fixed (for a given choice of parameters, see [38]).

Appendix B.1. Finite Element Quadrature for 2D Problems

For 2D problems we also used finite element quadrature to estimate the integrals and also to estimate the negative-order Sobolev norm. This approach does not scale to higher-dimensional problems, but is a good method for the purposes of validating algorithms based on the variational optimization and neural networks. Let $V_h \subset H^1_\pi(\Omega)$ be a finite element space and let W_h be another finite element space such that $V_h \subset W_h \subset H^1_\pi(\Omega)$. To V_h and W_h we associate standard interpolation operators $I_{V_h} : C(\Omega) \to V_h$ and $I_{W_h} : C(\Omega) \to W_h$. For a given continuous realization of the neural network $\psi_\theta = \Xi \circ R_{\theta,\rho}$, we compute:

$$\int_\Omega \|\nabla\psi_\theta\|^2 + V\psi_\theta^2 \approx \int_\Omega \|\nabla(I_{V_h}\psi_\theta)\|^2 + V(I_{V_h}\psi_\theta)^2$$

and we use standard finite element quadratures to evaluate the integrals on the right-hand side, see for instance the FEniCS book [48].

Finally, to assess the negative-order Sobolev norm of the residual we use the auxiliary subspace W_h and compute, using standard finite element calculus:

$$\sup_{\psi\in H^1_\pi(\Omega)} \frac{\left|\int_\Omega (\nabla\psi\cdot\nabla\phi + (V-\varepsilon(\psi))\psi\phi)\right|}{\|\phi\|} \approx \sup_{\psi\in W_h} \frac{\left|\int_\Omega (\nabla(I_{V_h}\psi)\cdot\nabla\phi + (V-\varepsilon(I_{V_h}\psi))(I_{V_h}\psi)\phi)\right|}{\|\phi\|} \,.$$

Appendix B.2. Direct Approximations for Higher Dimensional Problems

For higher-dimensional problems we use quadrature rules based on low discrepancy sequences [39] to compute the value of the energy functional (Rayleigh quotient) on the returned neural network realization $R_{\theta,\rho}$. To define the loss function, which consists of the energy functional and the normalization constraints, we use the quasi-Monte Carlo approach [49], but with fewer quadrature nodes. This is consistent with the approach taken in the original Deep Ritz algorithm. The use of Smolyak's rules can also be considered. This, however, does not lead to numerically stable optimization procedures. The realizations of neural networks are functions that can have many sharp local extrema and so optimizing using a fixed collection of nodes was observed to lead to degenerate solutions. The further problem stems from the fact that Smolyak's rules have, unlike Gaussian rules, a certain percentage of negative weights and so due to approximation errors it is possible to compute a negative approximation of an integral of a positive function. This is highly undesirable for a minimization procedure. Quasi-Monte Carlo integration routines are much less accurate than sparse grid rules, but all of their integration weights are positive and in addition they avoid the problem of overfitting. We use Sobol's points [39] to calculate nodes for the quasi-Monte Carlo approximation of the energy functional for higher-dimensional problems.

Appendix C. Architecture of the VPINN Neural Network

We will present the architectures of deep neural networks used in the paper. The network architecture $\vec{N}$ as defined in Definition 1 is sufficient to describe deep dense neural networks. Neural network architectures are presented graphically. Some further formal descriptions of the approximation classes can be found in [50]. The particular architecture that we use will have slightly more regularity in the dimensions of the layers. However, there will be more links between layers, which are inspired by the `DenseNet` concept [51]. The architecture is depicted in Figure A2 and we use the vector $\vec{N}_{\texttt{DenseNet}} = (n, k, l, m)$ to describe the architecture of the network, which has k blocks with l layers of the size m. Realizations of this network are functions from $\mathbb{R}^n$ to $\mathbb{R}$.

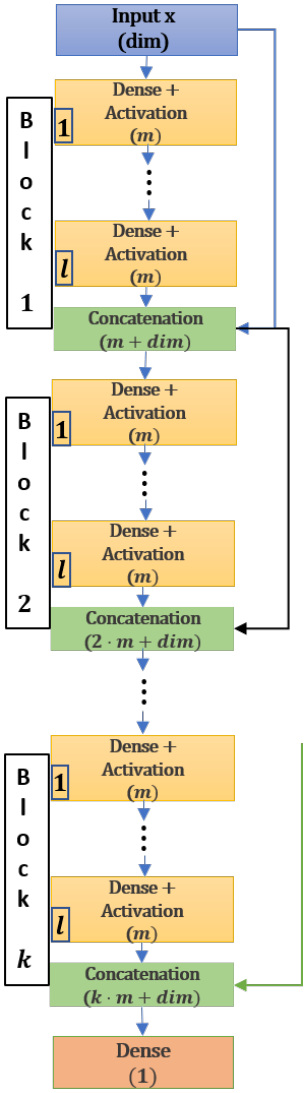


Figure A1. VPINN architecture with k blocks, l layers in each block and m neurons in each dense layer.

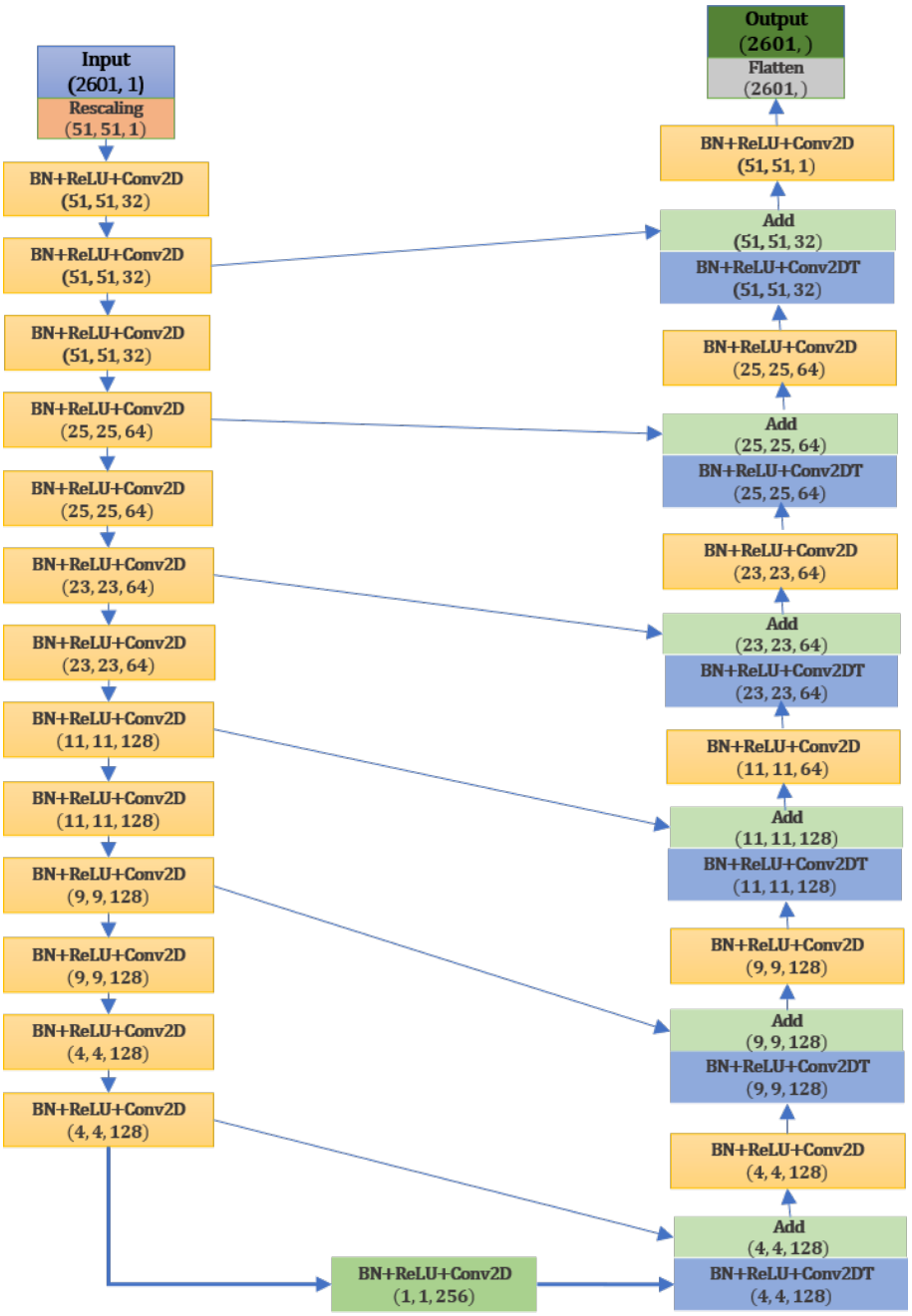


Figure A2. FCNN encoder–decoder architecture inspired by the U-Net concept from [30].

References

1. Reed, M.; Simon, B. *Methods of Modern Mathematical Physics, III*; Scattering Theory; Academic Press [Harcourt Brace Jovanovich, Publishers]: New York, NY, USA; London, UK, 1979.
2. Teschl, G. Mathematical methods in quantum mechanics. In *Graduate Studies in Mathematics*; With Applications to Schrödinger Operators; American Mathematical Society: Providence, RI, USA, 2009; Volume 99, pp. xiv+305.
3. Mills, K.; Spanner, M.; Tamblyn, I. Deep learning and the Schrödinger equation. *Phys. Rev. A* **2017**, *96*, 042113. [CrossRef]
4. Anderson, P.W. Absence of Diffusion in Certain Random Lattices. *Phys. Rev.* **1958**, *109*, 1492–1505. [CrossRef]
5. Arnold, D.N.; David, G.; Filoche, M.; Jerison, D.; Mayboroda, S. Computing spectra without solving eigenvalue problems. *SIAM J. Sci. Comput.* **2019**, *41*, B69–B92. [CrossRef]
6. Arnold, D.N.; David, G.; Jerison, D.; Mayboroda, S.; Filoche, M. Effective Confining Potential of Quantum States in Disordered Media. *Phys. Rev. Lett.* **2016**, *116*, 056602. [CrossRef] [PubMed]
7. Arnold, D.N.; David, G.; Filoche, M.; Jerison, D.; Mayboroda, S. Localization of eigenfunctions via an effective potential. *Comm. Partial. Differ. Equations* **2019**, *44*, 1186–1216. [CrossRef]

8. Khoromskij, B.N.; Oseledets, I.V. QTT approximation of elliptic solution operators in higher dimensions. *Russ. J. Numer. Anal. Math. Model.* **2011**, *26*, 303–322. [CrossRef]
9. Orús, R. A practical introduction to tensor networks: Matrix product states and projected entangled pair states. *Ann. Phys.* **2014**, *349*, 117–158. [CrossRef]
10. Raissi, M.; Perdikaris, P.; Karniadakis, G.E. Physics Informed Deep Learning (Part I): Data-driven Solutions of Nonlinear Partial Differential Equations. *arXiv* **2017**, arXiv:1711.10561.
11. Mishra, S.; Molinaro, R. Estimates on the generalization error of Physics Informed Neural Networks (PINNs) for approximating PDEs. *arXiv* **2020**, arXiv:2006.16144.
12. Lagaris, I.; Likas, A.; Fotiadis, D. Artificial neural network methods in quantum mechanics. *Comput. Phys. Commun.* **1997**, *104*, 1–14. [CrossRef]
13. Steinerberger, S. Localization of quantum states and landscape functions. *Proc. Am. Math. Soc.* **2017**, *145*, 2895–2907. [CrossRef]
14. Hermann, J.; Schätzle, Z.; Noé, F. Deep-neural-network solution of the electronic Schrödinger equation. *Nat. Chem.* **2020**, *12*, 891–897. [CrossRef] [PubMed]
15. Graziano, G. Deep learning chemistry ab initio. *Nat. Rev. Chem.* **2020**, *4*, 564. [CrossRef]
16. Han, J.; Jentzen, A.; Weinan, E. Solving high-dimensional partial differential equations using deep learning. *Proc. Natl. Acad. Sci. USA* **2018**, *115*, 8505–8510. [CrossRef]
17. Han, J.; Zhang, L.; Weinan, E. Solving many-electron Schrödinger equation using deep neural networks. *J. Comput. Phys.* **2019**, *399*, 108929. [CrossRef]
18. Beck, C.; Weinan, E.; Jentzen, A. Machine learning approximation algorithms for high-dimensional fully nonlinear partial differential equations and second-order backward stochastic differential equations. *J. Nonlinear Sci.* **2019**, *29*, 1563–1619. [CrossRef]
19. Ma, C.; Wang, J.; Weinan, E. Model reduction with memory and the machine learning of dynamical systems. *Commun. Comput. Phys.* **2019**, *25*, 947–962. [CrossRef]
20. Weinan, E.; Yu, B. The Deep Ritz method: A deep learning-based numerical algorithm for solving variational problems. *Commun. Math. Stat.* **2018**, *6*, 1–12.
21. Kharazmi, E.; Zhang, Z.; Karniadakis, G.E. Variational Physics-Informed Neural Networks For Solving Partial Differential Equations. *arXiv* **2019**, arXiv:1912.00873.
22. Zhang, L.; Han, J.; Wang, H.; Saidi, W.; Car, R.; Weinan, E. End-to-end Symmetry Preserving Inter-atomic Potential Energy Model for Finite and Extended Systems. In *Advances in Neural Information Processing Systems*; Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., Garnett, R., Eds.; Curran Associates, Inc.: Red Hook, NY, USA, 2018; Volume 31, pp. 4436–4446.
23. Weinan, E.; Han, J.; Zhang, L. Integrating Machine Learning with Physics-Based Modeling. *arXiv* **2020**, arXiv:2006.02619.
24. McFall, K.S.; Mahan, J.R. Artificial Neural Network Method for Solution of Boundary Value Problems With Exact Satisfaction of Arbitrary Boundary Conditions. *IEEE Trans. Neural Netw.* **2009**, *20*, 1221–1233. [CrossRef] [PubMed]
25. Kato, T. *Perturbation Theory for Linear Operators*; Classics in Mathematics; Reprint of the 1980 Edition; Springer: Berlin, Germany, 1995; p. xxii+619.
26. Kato, T. On the upper and lower bounds of eigenvalues. *J. Phys. Soc. Jpn.* **1949**, *4*, 334–339. [CrossRef]
27. Grubišić, L. On eigenvalue and eigenvector estimates for nonnegative definite operators. *SIAM J. Matrix Anal. Appl.* **2006**, *28*, 1097–1125. [CrossRef]
28. Grubišić, L.; Ovall, J.S. On estimators for eigenvalue/eigenvector approximations. *Math. Comp.* **2009**, *78*, 739–770. [CrossRef]
29. Hesthaven, J.S.; Rozza, G.; Stamm, B. *Certified Reduced Basis Methods for Parametrized Partial Differential Equations*; SpringerBriefs in Mathematics; BCAM SpringerBriefs; Springer: Cham, Switzerland; BCAM Basque Center for Applied Mathematics: Bilbao, Spain, 2016.
30. Ronneberger, O.; Fischer, P.; Brox, T. U-Net: Convolutional Networks for Biomedical Image Segmentation. In *Medical Image Computing and Computer-Assisted Intervention— MICCAI 2015*; Navab, N., Hornegger, J., Wells, W.M., Frangi, A.F., Eds.; Springer International Publishing: Cham, Switzerland, 2015; pp. 234–241.
31. Müller, J.; Zeinhofer, M. Deep Ritz revisited. *arXiv* **2020**, arXiv:1912.03937.
32. Golub, G.H.; Van Loan, C.F. *Matrix Computations*, 4th ed.; Johns Hopkins Studies in the Mathematical Sciences; Johns Hopkins University Press: Baltimore, MD, USA, 2013.
33. Arora, R.; Basu, A.; Mianjy, P.; Mukherjee, A. Understanding Deep Neural Networks with Rectified Linear Units. *arXiv* **2018**, arXiv:1611.01491.
34. Grubišić, L.; Nakić, I. Error representation formula for eigenvalue approximations for positive definite operators. *Oper. Matrices* **2012**, *6*, 793–808. [CrossRef]
35. Bank, R.E.; Grubišić, L.; Ovall, J.S. A framework for robust eigenvalue and eigenvector error estimation and Ritz value convergence enhancement. *Appl. Numer. Math.* **2013**, *66*, 1–29. [CrossRef]
36. Davis, C.; Kahan, W.M. The rotation of eigenvectors by a perturbation. III. *SIAM J. Numer. Anal.* **1970**, *7*, 1–46. [CrossRef]
37. Kingma, D.P.; Ba, J. Adam: A Method for Stochastic Optimization. *arXiv* **2015**, arXiv:1412.6980.
38. Feinberg, J.; Langtangen, H.P. Chaospy: An open source tool for designing methods of uncertainty quantification. *J. Comput. Sci.* **2015**, *11*, 46–57. [CrossRef]

39. Sobol, I.M. Distribution of points in a cube and approximate evaluation of integrals. *Ž. Vyčisl. Mat. Mat. Fiz.* **1967**, *7*, 784–802. [CrossRef]
40. Smoljak, S.A. Quadrature and interpolation formulae on tensor products of certain function classes. *Dokl. Akad. Nauk SSSR* **1963**, *148*, 1042–1045.
41. Mishra, S.; Molinaro, R. Estimates on the generalization error of Physics Informed Neural Networks (PINNs) for approximating PDEs II: A class of inverse problems. *arXiv* **2020**, arXiv:2007.01138.
42. Abadi, M.; Agarwal, A.; Barham, P.; Brevdo, E.; Chen, Z.; Citro, C.; Corrado, G.S.; Davis, A.; Dean, J.; Devin, M.; et al. TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems. *arXiv* **2016**, arXiv:1603.04467.
43. Platte, R.B.; Trefethen, L.N. Chebfun: A new kind of numerical computing. In *Progress in industrial mathematics at ECMI 2008*; Springer: Heidelberg, Germany, 2010; Volume 15, pp. 69–87.
44. Trefethen, L.N. *Approximation Theory and Approximation Practice*; Society for Industrial and Applied Mathematics (SIAM): Philadelphia, PA, USA, 2013.
45. Han, J.; Jentzen, A. Algorithms for Solving High Dimensional PDEs: From Nonlinear Monte Carlo to Machine Learning. *arXiv* **2020**, arXiv:2008.13333.
46. Kazeev, V.; Oseledets, I.; Rakhuba, M.; Schwab, C. QTT-finite-element approximation for multiscale problems I: model problems in one dimension. *Adv. Comput. Math.* **2017**, *43*, 411–442. [CrossRef]
47. Chollet, F. Keras. 2015 Available online: https://keras.io (accessed on 7 January 2021).
48. Logg, A.; Mardal, K.A.; Wells, G.N. *Automated Solution of Differential Equations by the Finite Element Method*; Springer: Berlin/Heidelberg, Germany, 2012.
49. Sobol, I.M.; Shukhman, B.V. QMC integration errors and quasi-asymptotics. *Monte Carlo Methods Appl.* **2020**, *26*, 171–176. [CrossRef]
50. Gribonval, R.; Kutyniok, G.; Nielsen, M.; Voigtlaender, F. Approximation spaces of deep neural networks. *arXiv* **2020**, arXiv:1905.01208.
51. Huang, G.; Liu, Z.; van der Maaten, L.; Weinberger, K.Q. Densely Connected Convolutional Networks. *arXiv* **2018**, arXiv:1608.06993.