Inverting the Kohn–Sham equations with physics-informed machine learning

IOP Publishing Mach. Learn.: Sci. Technol. 5 (2024) 015050 https://doi.org/10.1088/2632-2153/ad3159

MACHINE LEARNING
Science and Technology

OPEN ACCESS

RECEIVED
15 December 2023

REVISED
7 February 2024

ACCEPTED FOR PUBLICATION
7 March 2024

PUBLISHED
19 March 2024

PAPER

Inverting the Kohn–Sham equations with physics-informed machine learning

Vincent Martinetto[1,4], **Karan Shah**[2,3,4], **Attila Cangi**[2,3,*] and **Aurora Pribram-Jones**[1,*]

[1] Department of Chemistry and Biochemistry, University of California Merced, 5200 North Lake Rd., Merced, CA 95343, United States of America
[2] Center for Advanced Systems Understanding, 02826 Görlitz, Germany
[3] Helmholtz-Zentrum Dresden-Rossendorf, 01328 Dresden, Germany
[4] These authors have contributed equally.
[*] Authors to whom any correspondence should be addressed.

E-mail: a.cangi@hzdr.de and apj@ucmerced.edu

Keywords: physics-informed neural networks, density-to-potential inversions, density functional theory, neural operators

Supplementary material for this article is available online

Abstract

Electronic structure theory calculations offer an understanding of matter at the quantum level, complementing experimental studies in materials science and chemistry. One of the most widely used methods, density functional theory, maps a set of real interacting electrons to a set of fictitious non-interacting electrons that share the same probability density. Ensuring that the density remains the same depends on the exchange-correlation (XC) energy and, by a derivative, the XC potential. Inversions provide a method to obtain exact XC potentials from target electronic densities, in hopes of gaining insights into accuracy-boosting approximations. Neural networks provide a new avenue to perform inversions by learning the mapping from density to potential. In this work, we learn this mapping using physics-informed machine learning methods, namely physics informed neural networks and Fourier neural operators. We demonstrate the capabilities of these two methods on a dataset of one-dimensional atomic and molecular models. The capabilities of each approach are discussed in conjunction with this proof-of-concept presentation. The primary finding of our investigation is that the combination of both approaches has the greatest potential for inverting the Kohn–Sham equations at scale.

1. Introduction

Modern, high-performance computational resources have enabled large-scale electronic structure simulations of molecules, materials, and other systems of interest across biology, chemistry, physics, and beyond. Kohn–Sham (KS) density functional theory (DFT) [1, 2] is the most widely used method due to its accuracy and computational efficiency. KS-DFT has helped solve major scientific and technological problems such as simulating chemical reactions, computing material properties, finding new catalysts, discovering drugs, and modeling microscopic environmental processes.

The electronic structure problem is typically tackled by solving the non-relativistic Schrödinger equation for the molecular Hamiltonian within the Born-Oppenheimer approximation. KS-DFT simplifies this process by transforming the many-body problem into an effective single-particle problem, utilizing the one-to-one correspondence of the external potential and the electronic density [1]. The crux of KS-DFT is producing an electronic density that matches the one obtained from solving the interacting many-body problem [2]. This is accomplished through an effective single-particle potential known as the KS potential, which is comprised of the external potential, the Hartree potential that accounts for electrostatic repulsion, and an exchange-correlation (XC) potential compensating for all interactions beyond the Hartree potential.

While an exact expression for the XC potential ($v_{\mathrm{XC}} = \delta E_{\mathrm{XC}}/\delta n$) is not known, useful approximations exist. These approximations are categorized based on increasing complexity and accuracy [3]. They include, for example, the local density approximation [2, 4, 5], which is derived from the interacting uniform electron gas [6]; generalized gradient approximations (GGAs) [4, 7–9], relying on density gradients; meta-GGAs [10, 11], which consider the kinetic energy density; and hybrid functionals, [4, 12, 13] incorporating a dependence on Hartree–Fock orbitals. More sophisticated approximations beyond hybrid functionals include the random phase approximation [14–19] and double hybrid [20–23] methods.

Despite the usefulness of existing approximations for various applications in chemistry and materials science, developing more accurate XC approximations is an active area of research. [24] New paths for finding and optimizing such approximations are required for describing complex molecular systems of upcoming interest [25]. Some advances in functional construction leverage machine learning (ML) techniques [26, 27]. These methods share the common goal of learning the XC energy functional from accurate data on molecular systems [28–51].

Another promising approach to constructing approximations to the XC functional is by reverse-engineering the XC potential from systems that can be solved exactly. This procedure takes the electronic density as input and finds the corresponding XC potential by inverting the KS equations. Though straightfoward to describe, this process is an example of an inverse problem. Inverse problems are encountered in various scientific fields, such as image reconstruction, a fundamental method in MRI and CT scans; inverse scattering, commonly used in seismology and radar imaging; and tomography, frequently employed in CT scans. In each case, as in our case, observed data is used to reconstruct the structure of an object that gives rise to the initial data [52]. Inverse problems are often ill-posed, meaning they are highly sensitive to small changes in the observed data and noise. This sensitivity makes them challenging to solve [53].

Various algorithmic and numerical schemes for inverting the KS equations exist in the literature [54–60]. These have provided valuable insights [61–66] into the properties of the XC potential, such as the appearance of the derivative discontinuity or the dynamical step structures in time-dependent DFT, and have guided the construction of XC approximations [67]. However, traditional inversion methods are often plagued by the numerical errors and instabilities [55] common in the solution of inverse problems, as well as from those specific to the density-to-potential problem.

Recent research has demonstrated numerous examples of utilizing ML, particularly neural networks, to address inverse problems, with a focus on medical image processing applications. The primary objective is to learn stable mappings that can be effectively applied to noisy data in a general context [68, 69]. The fusion of physics with ML offers innovative solutions to complex problems, leading to the evolution of physics-informed machine learning (PIML) techniques [70]. Instead of solely relying on data, PIML incorporates governing physical laws, typically differential equations, into the learning process. This results in data-efficient learning and equips the model with the capability to generalize within physically consistent bounds. This is especially relevant in cases with sparse or noisy data where conventional ML methods might not perform well due to overfitting or by producing non-physical outcomes.

In this work, we employ two recent advances in PIML, namely physics-informed neural networks (PINNs) [71] and Fourier neural operators (FNOs) [72], to tackle the inverse problem in KS-DFT. PINNs have the unique ability to learn from the underlying physics described by partial differential equations, leading to improved generalizability with less data and training time compared to conventional methods. This characteristic proves highly advantageous in computational sciences where data availability is limited and generating extensive training sets is costly. PINNs have demonstrated their effectiveness and versatility in various scientific domains for inversions, including materials and fluid dynamics models [73–75].

Neural operators (NOs) are a class of models that map function-to-function spaces, as opposed to the finite-dimensional vector space mappings of neural networks. This is especially useful for inverse PDE problems where experimental or simulation data is available. FNOs are a type of NO which represent operator weights in Fourier space. The main advantages of FNOs are that they generalize well across function spaces, and being resolution-invariant, they can be used for inference on higher resolution grids than the training set grids. FNOs have also been used for forward and inverse PDE problems in various domains [72, 76].

In the following, we construct both a PINN and an FNO to predict the XC potential based on input electronic densities. We demonstrate the implementation of these computational workflows and compare both approaches in terms of efficiency and accuracy for their applicability as KS inverters. We observe higher levels of accuracy and ease of use for FNOs and more predictable errors and greater extrapolation power for PINNs.

2. Methods

2.1. KS DFT

In the framework of KS-DFT, we solve a set of effective single-particle equations known as the KS equations (which we present here using atomic units [77] for clarity and simplicity of notation),

$$\left[-\frac{\nabla^2}{2}+v_s(\boldsymbol{r})\right]\phi_i(\boldsymbol{r})=\epsilon_i\ \phi_i(\boldsymbol{r})\ , \tag{1}$$

where $\phi_i(\boldsymbol{r})$ represents the KS eigenfunctions and ϵ_i denotes the KS eigenvalues. These KS equations constitute an auxiliary system designed to replicate the electron density of an interacting many-body system, expressed as

$$n(\boldsymbol{r})=\sum_{i=1}^{N}|\phi_i(\boldsymbol{r})|^2\,, \tag{2}$$

where the summation index corresponds to the total number of electrons, denoted as N.

The correspondence of the density defined in equation (2) with the density of an interacting many-body system is established through the KS potential which mimics the inter-electronic interaction and is defined as

$$v_{\rm S}(\boldsymbol{r})=v_{\rm ext}(\boldsymbol{r})+v_{\rm H}(\boldsymbol{r})+v_{\rm XC}(\boldsymbol{r})\ . \tag{3}$$

Here, $v_{\rm ext}(\boldsymbol{r})$ is the external potential, $v_{\rm H}(\boldsymbol{r})$ is the Hartree potential (i.e. the electrostatic potential due to an electronic density interacting with itself) and $v_{\rm XC}(\boldsymbol{r})$ is the XC potential. Both the external and Hartree potentials are known exactly in terms of the density. The only unknown is the XC potential. While solving the set of KS equations with the exact XC potential yields the exact density of the many-body system, in practice, approximations to the XC energy need to be used. A plethora of approximations is available in the literature [2, 4–13] and more accurate approximations are continuously being developed.

Usually, the KS equations are solved iteratively. This means that a system is defined and an XC approximation is selected. An initial guess of the ground-state density is made, which in turn provides guesses for the Hartree and XC potentials. Equations (1) and (2) are solved, generating a new density, which can be used to calculate new Hartree and XC potentials. This cycle will continue until the density converges to within a chosen criterion. In the end, the resulting density is an accurate approximation of the interacting many-body system, which can be used to calculate electronic properties.

Conversely, equations (1) and (2) can be solved to find the exact KS potential that results in a given target density. In this case, no approximation is made to the XC potential, only an initial guess. Then, by solving equations (1) and (2), the guess for the XC potential can be updated. This process is much less stable than the 'direct' one described above because the potential is sensitive to small changes in the density.

We use the iDEA code [78, 79] for generating data and as benchmark for inversion. In iDEA, the density-to-potential inversion is done iteratively:

$$v_s\rightarrow v_s+\mu\left[n(\boldsymbol{r})^p-n_0(\boldsymbol{r})^p\right]\ , \tag{4}$$

where $n_0(r)$ represents the target density, while μ and p serve as convergence parameters.

2.2. Model system

Model systems in one dimension are a useful tool for the development of DFT approximations [59] for two reasons. First, it is simpler to obtain highly accurate or exact data from a one-dimensional system. Second, these model systems can be finely tuned to mimic a wide range of well-understood physical phenomena. Therefore, we leverage one-dimensional models to demonstrate the utility of our inversion method. We define two model systems: a one-dimensional atom and a one-dimensional diatomic molecule, as illustrated in figure 1. A one-dimensional atom is defined by the external potential

$$v_{\rm ext}(\boldsymbol{r})=-\frac{Z}{|\boldsymbol{r}|+a}\,, \tag{5}$$

where Z is the charge of the atomic well and a is softening parameter to ensure that the potential is defined at all points in space. This simple model can readily be extended to a one-dimensional model of a diatomic molecule by incorporating two potential wells through an additional parameter, d, which represents the bond length in terms of the separation distance between the two wells:

$$v_{\rm ext}(\boldsymbol{r})=-\frac{Z_1}{|\boldsymbol{r}-\frac{d}{2}|+a}-\frac{Z_2}{|\boldsymbol{r}+\frac{d}{2}|+a}. \tag{6}$$

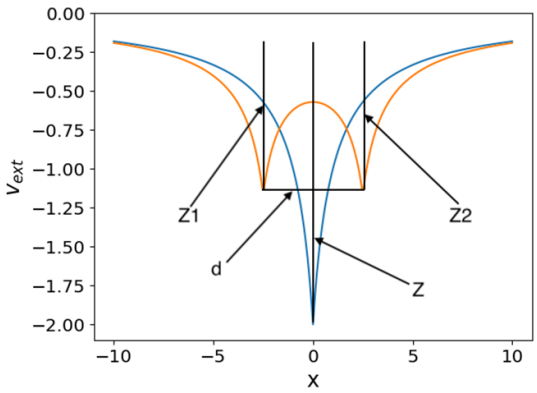


Figure 1. Illustration of the model systems: one-dimensional atom (blue) and a one-dimensional diatomic molecule (orange) with variable parameters charge Z, Z_1, and Z_2 and bond length d.

This idea can be extended by defining additional wells to move from an atomistic view to a material or crystalline structure in one dimension.

All data used to train the presented models were generated using external potentials defined in this manner. The electronic densities and KS eigenfunctions were generated using the iDEA code [78, 79]. For the one-dimensional atom of varying charge, Z was varied between the values of two and six in steps of 0.1, resulting in a dataset with 41 individual data points which was split into a train, test, and validation set. These sets had 32, 5, and 4 individual data points each, respectively. For the one-dimensional diatomic molecule, Z_1, Z_2, and d were all varied within the range of one to five, in steps of 0.5. All combinations of the three variables were generated, resulting in a data set with 729 data points. The dataset was split such that the training set had 590 points, the test set had 73, and the validation set had the remaining 66. The dataset can be downloaded from the HZDR Data Repository Rodare via the link https://rodare.hzdr.de/record/2720 [80].

2.3. Physics-informed neural networks

As a subset of neural networks, PINNs use equations and conditions of the underlying physics to enable a neural network to predict physically relevant information. Consider a general partial differential equation of the form

$$u(x,t) + \mathcal{N}[u;\lambda] = 0\,, \tag{7}$$

where u is the solution, $x \in \Omega$ a D-dimensional spatial coordinate with Ω a subset of $\mathbb{R}^D$, $t \in [0,T]$ the time variable, $\mathcal{N}$ a nonlinear operator, and λ the set of parameters that describe a physical system. If a neural network is used to represent the solution u, as $\tilde{u}$, this gives rise to

$$\tilde{u}(x,t) + \mathcal{N}[\tilde{u};\lambda] := f(x,t)\ . \tag{8}$$

The function f must only equal zero when the network accurately represents the solution to the differential equation for the given parameters λ. If trained on a wide set of physical data, it should be able to predict solutions to the differential equation within that set more generally.

Based on equation (1), we derive the following model and loss functions within the framework of PINNs:

$$f(\boldsymbol{r}) = \epsilon_i\ \phi_i(\boldsymbol{r}) + \frac{\nabla^2}{2}\ \phi_i(\boldsymbol{r}) - v_s(\boldsymbol{r})\ \phi_i(\boldsymbol{r}) \tag{9}$$

$$L_{\text{PDE}} = MSE(f,0) = \frac{1}{N_f}\sum_{i=1}^{N_f}\left|f\left(x_f^i\right)\right|^2\,, \tag{10}$$

where the loss function L_{PDE} is computed in terms of a mean squared error (MSE) $MSE(f,0)$, with x_f^i denoting one of the N_f collocation points where the solution is provided.

Here, our focus is to invert the KS equation, wherein we have the set of solutions, $\{\phi_i\}$, and our goal is to get the network to predict the parameters, v_s, that lead to an appropriate set of solutions. If the network can correctly predict v_s, then the function f should return a value of zero. What distinguishes this approach from conventional data-driven ML is that the network does not receive the solution set as input for prediction. Instead, it receives only the weighted sum of the solutions in the form of the electronic density calculated using equation (2).

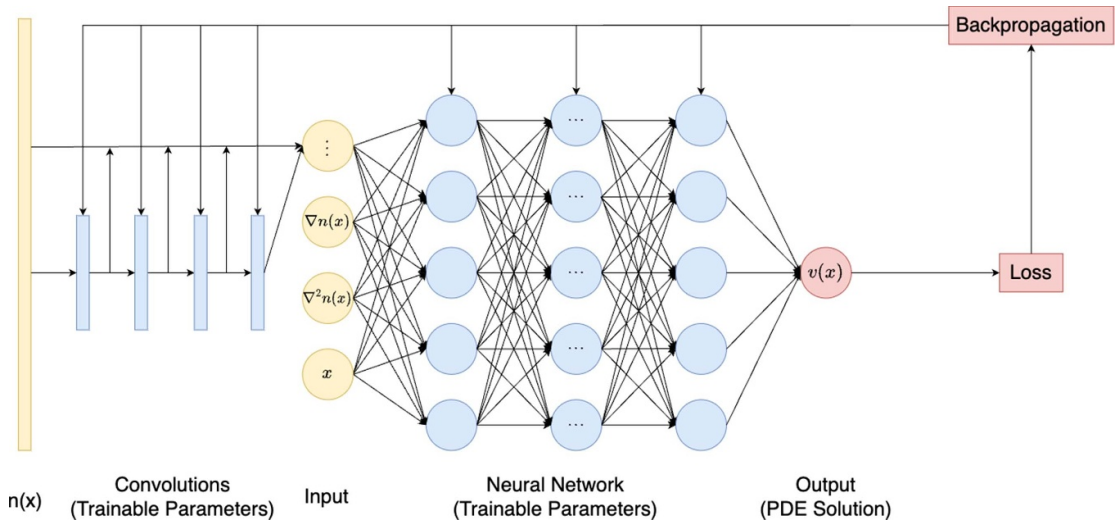


Figure 2. PINN architecture for predicting the KS potential based on an input electronic density.

In figure 2, we illustrate our implementation of a PINN that predicts the KS potential for an input density. The network takes the density as an input and processes it through a series of convolutions. The output of each convolution as well as the initial density is used as an input into a dense network. The network is then asked to predict the KS potential on each point in space given this set of features, as well as the first and second derivative of the density.

Convolutional neural networks are a common ML technique for image recognition [81–84]. They observe both local and non-local features in input data to predict what is present in an image. The convolution of two functions f and g is defined as

$$(f*g)(t) = \int f(\tau)\, g(t-\tau)\, d\tau \tag{11}$$

and can be described as sum of the overlap of two functions at each point in t. In our convolutional PINN, this is accomplished by using a set of convolutions in series and in parallel. Back propagation is used to learn functions g that emphasize input features crucial for predicting a class. Increasing the number of convolutions in series results in the passing of additional non-local information to the features.

It is widely known [1, 4] that the XC potential is a nonlocal functional of the electronic density. Consequently, training a network to correctly predict an XC potential based only on the electronic density at a single point in space is ill-advised. Our PINN model uses convolutions to account for some of the non-locality of the XC potential. Each step through the convolutions indicates a larger regime of non-locality that has been included in its output features.

2.4. FNOs

Neural operators (NOs) [85, 86] are an extension of neural networks that take functions as inputs and return functions as outputs. While traditional neural networks map finite-dimensional vectors to finite-dimensional vectors, neural operators work on infinite-dimensional function spaces. A neural network can be defined as a mapping $f: \mathbb{R}^n \to \mathbb{R}^m$, whereas a neural operator maps functions to functions, i.e. $G: \mathcal{A} \to \mathcal{U}$, where $\mathcal{A}$ and $\mathcal{U}$ are function spaces. Given that there exists a non-linear map $\mathcal{G}^\dagger : \mathcal{A} \to \mathcal{U}$, our goal is to construct a neural operator that approximates this map, i.e. $\mathcal{G}_\theta \approx \mathcal{G}^\dagger$, where $\mathcal{G}_\theta : \mathcal{A} \to \mathcal{U}$ with parameters θ in finite dimensional space $\mathbb{R}^p$.

The neural operator is trained on point-wise (in function space) observations of the form $\{(a_i, u_i)\}_{i=1}^N$, where $(a_i \in \mathcal{A}; a_i(x) \in \mathbb{R}, x \in \mathbb{R})$, $(u_i \in \mathcal{U}; u_i(x) \in \mathbb{R}, x \in \mathbb{R})$ and $u_i = \mathcal{G}^\dagger(a_i)$. The goal is to find the parameters θ^* that minimize the loss

$$\theta^* = \min_{\theta \in \mathbb{R}^p} \frac{1}{N} \sum_{i=1}^{N} \left\| u^{(i)} - \mathcal{G}_\theta(a_i) \right\|_{\mathcal{U}}^2. \tag{12}$$

Analogous to the neural networks being defined as multi-layer compositions of linear and non-linear operations, we can define a neural operator $\mathcal{G}_\theta(a)$ as multi-layer compositions of linear and non-linear operators acting directly in function space:

$$\mathcal{G}_\theta(a) = Q(v_L(v_{L-1}(\dots v_1(P(a))))) , \tag{13}$$

where the layer v_{l+1} is defined as

$$v_{l+1}(x) = \sigma_{(l+1)}(W_l v_l(x) + (\mathcal{K}_l(a;\lambda)\, v_l)(x)) . \tag{14}$$

Here, the non-local kernel integral operator $\mathcal{K}_l$ is defined as

$$(\mathcal{K}_l(a;\lambda)\, v_l)(x) = \int_{\Omega_l} \kappa_l(x, y, a(x), a(y))\, v_l(y)\, dy \tag{15}$$

over the domain Ω_l.

The kernel function $\kappa_l(x, y, a(x), a(y))$ is a function of x, y, and the input functions $a(x)$ and $a(y)$. The kernel function, parametrized by λ, and the local linear operator W_l, a matrix of weights, are learned during training. $\sigma_{(l+1)}$ is a local non-linear activation function. The local operator $P(a)$ is a pre-processing operator that transforms the input function a into a higher dimensional representation v_0, which is the input to the first layer. The local operator $Q(v_L)$ is a post-processing operator that projects the output of the last layer v_L back into $\mathcal{U}$.

Several methods can be used to approximate the global integral calculation in each layer, such as graph neural networks with Monte-Carlo sampling [86] and multipole graph neural operators [87]. These approximations reduce the computational complexity and allow for scalable solutions [76].

FNOs [72] leverage the Fourier transform to work in the frequency domain. Notice that by using a kernel $\kappa(x-y)$, equation (15) is a convolution operator, and by using the convolution theorem, equation (15) becomes

$$(\mathcal{K}_l(a;\lambda)\, v_l)(x) = \mathcal{F}^{-1}(\mathcal{F}(\kappa_l) \cdot \mathcal{F}(v_l))(x) \tag{16}$$

$$\mathcal{F}(x) = \sum_{n=0}^{N-1} x_n \cdot e^{-\frac{i2\pi}{N} kn} = X_k \tag{17}$$

$$\mathcal{F}^{-1}(X) = \frac{1}{N} \sum_{n=0}^{N-1} X_k \cdot e^{\frac{i2\pi}{N} kn} = x_n, \tag{18}$$

where $\mathcal{F}$ is the Fourier transform and $\mathcal{F}^{-1}$ is its inverse. The kernel κ_l is parametrized in Fourier space. Here the parameterization is finite-dimensional determined by the number of Fourier modes $k_{\max}$. By working in frequency domain, the kernels can be computed efficiently using the fast-Fourier tranform and the model can capture global patterns more effectively.

A distinction has to be made between the solving approach (PINNs) and the learning approach (NOs) for PDE problems. PINNs are useful for obtaining numerical solutions for specific initial and boundary conditions and PDE parametrizations. In contrast, learning a PDE via neural operators aims to learn the underlying operator itself, enabling the prediction of solutions for various conditions without re-solving the PDE. PINNs can be used to solve PDEs if the PDEs are defined, even in the absence of training data, while NOs can be used to approximate undefined PDEs with training data. NOs are resolution-invariant and can also be used for zero-shot super-resolution [72]. A rigorous mathematical definition and analysis for NOs is provided in [76]. The strengths of FNOs in this context lie in their inherent capability to capture non-local dependencies and their resolution invariance.

Given the non-local dependence of the XC potential on the electronic density, the non-locality of FNO layers make them well-suited for the inverse problem in KS-DFT. The input in this case is a grid of electronic density values and the output is the XC potential at the corresponding spatial grid points. The loss function is simply the MSE between true and predicted XC potential:

$$L_{\text{FNO}} = \text{MSE}(v_{\text{XC}}, \tilde{v}_{\text{XC}}) = \frac{1}{N} \sum_{i=1}^{N} \left| v_{\text{XC}}^{(i)} - \tilde{v}_{\text{XC}}^{(i)} \right|^2 \tag{19}$$

where $\tilde{v}_{\text{XC}}$ denotes the predicted XC potential for system i. The architecture we implemented is shown in figure 3.

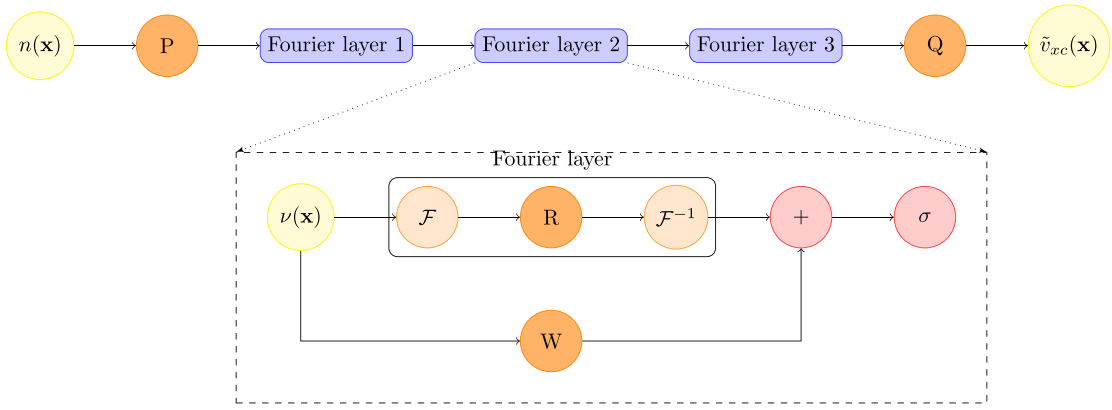


Figure 3. FNO architecture for predicting the KS potential based on an input electronic density.

2.5. Similarity measures

We evaluate the complexity of the datasets using two different similarity measures, namely the cosine similarity and Euclidean distance. The cosine similarity between two vectors $\mathbf{X}$ and $\mathbf{Y}$ in $\mathbb{R}^D$ is defined as

$$\cos(\theta) = \frac{\mathbf{X}\cdot\mathbf{Y}}{\|\mathbf{X}\|\|\mathbf{Y}\|}, \tag{20}$$

which returns values in the range $[-1, 1]$, namely -1 if the vectors point in opposite directions, 1 if they point in the same direction, and 0 if they are orthogonal to each other. The Euclidean distance between the same vectors is defined as

$$D(\mathbf{X}, \mathbf{Y}) = \|\mathbf{X} - \mathbf{Y}\|. \tag{21}$$

This measure has a lower bound of zero but no upper limit. The value returned is 0 only if the vectors $\mathbf{X}$ and $\mathbf{Y}$ are the same vector. Then, as two vectors continue to move further apart, the value moves further away from zero. The Euclidean distance includes some details related to the cosine similarity, but it also describes the difference in magnitude of the values found in the two vectors. This is a significant piece of information that the cosine similarity measure misses. Both of these measures provide insight into the complexity of each dataset and the network architecture's capacity to work on each of them.

We illustrate the similarity for both the atomic and the molecular model systems in figure 4. As expected, the diatomic system is more complex than the atomic system, which is evident from the broader range of values in the diatomic case, versus the atomic case, for both cosine similarity and Euclidean distance measures. In the cosine similarity plots for the diatomic system, a block pattern emerges. The largest blocks shift with a change in the bond length. Descending to the bottom or moving to the right of the plot corresponds to an increase in the separation between two atomic centers. The greatest dissimilarity is noted at the two extremes of separation. Interestingly, at larger separations, the vectors show slightly less self-similarity compared to those at lower separations, as indicated by the darker shading of the bottom right block compared to the top left. The Euclidean distance plot yields similar trends, although it does not show this distinction within varying separation distances.

3. Results

We next present our findings for the two PIML approaches, comparing results using the PINN model as well as the FNO model. We trained both models on the same datasets and evaluate the model performance via the accuracy of the returned eigenvalues when solving the KS equations with the predicted XC potentials. We assess the ability of both PIML approaches as inverters for the KS equations in terms of the two aforemorentioned model systems: the one-dimensional atom with varying charge and the diatomic molecule with varying charges and bond length. Both the PINN model and the FNO model were trained on model systems with a grid resolution of 301 in real space. The performance was measured for systems with resolutions of 301 and 501, as well as for 301-grid systems with parameters outside the training range (extrapolation).

In all model systems, we consider the first six eigenstates, where each model is occupied with two same-spin fermions. Additionally, we consider four virtual KS orbitals. The set of occupied and virtual

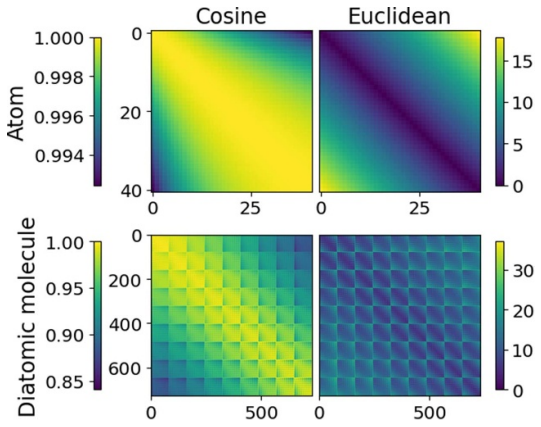


Figure 4. Similarity measures — cosine similarity (left) and Euclidean distance (right) — of the two model systems. The top panel shows the similarity measures for the one-dimensional atom with varying charge, while the bottom panel shows the similarity measures for the one-dimensional diatomic molecule with varying charge and bond length. The similarity measures illustrate the greater degree of complexity in the molecular model system. The axes are labeled by the system index.

eigenstates defines the shape of the KS potential. The lowest-energy eigenvalue indicates how well networks predict the cusp of the atoms, while the highest value signifies how well they predict the tail of the potential.

We evaluate each model for ten neural networks with different random seeds. The error is quantified using the mean absolute error (MAE) and the mean absolute percentage error (MAPE) of the eigenvalues defined as

$$\mathrm{MAE}_{i,j} = \frac{1}{N}\sum_{n=1}^{N}\left|\tilde{\epsilon}_{i,j,n} - \epsilon_{i,j}\right| \tag{22}$$

$$\mathrm{MAPE}_{i,j} = \frac{1}{N}\sum_{n=1}^{N}\left|\frac{\tilde{\epsilon}_{i,j,n} - \epsilon_{i,j}}{\epsilon_{i,j}}\right| \times 100\%\,, \tag{23}$$

where $\tilde{\epsilon}_{i,j,n}$ denotes the predicted eigenvalues and each predicted eigenvalue is indexed by i, j, n, with i denoting the level index, j the system index, and n the network index, with each neural network having a different random seed.

3.1. Inversions with physics-informed neural networks

3.1.1. Atom with varying charge

Figure 5 illustrates the performance of the PINN model for a one-dimensional atom with a variable parameter Z corresponding to its charge, plotting the MAE for a set of fixed charges. Each data point represents the mean of ten neural network predictions that were generated using varying network initializations and random seeds. The data points are color-coded to indicate ground and excited states in the order of blue, orange, red, green, purple, and brown. This order is maintained in the following discussion of the results.

The performance of the PINN model is best when the well charge Z is small. The error systematically increases with the charge, particularly for the lowest-lying eigenvalue. The overall MAE averaged over the test systems is 1.77×10^{-3} Ha (denoted by a gray horizontal line in figure 5). The PINN model is competitive with the chemical accuracy criterion of being within 1 kcal mol^{-1} or 0.0016 Ha, as marked by the red horizontal line in our figures.

We have also displayed the the MAPE of the individual eigenvalues in figure 6. The greatest error is in the first two eigenvalues of the systems, with both lying above the total average MAPE of around 0.13%, while the next four lie below this threshold.

We have also explored the relationship between the performance of the PINN model and the resolution of the spatial grid. We suspect that the source of the errors observed in figure 5 is related to the resolution of the grid used to train the PINN model, which depends on the accuracy of the KS orbitals and their second derivatives. We anticipate that as the grid resolution improves, the accuracy of the second derivatives will also improve, particularly at the sharp cusp-like feature in the KS potential. Our findings indicate that errors diminish with the enhancement of grid resolution. This correlation presents a methodical approach for reducing the error in the PINN model. The improvements are reported in table 1, which showcases the systematic enhancement in performance with increasing grid resolution (see Atom 501).

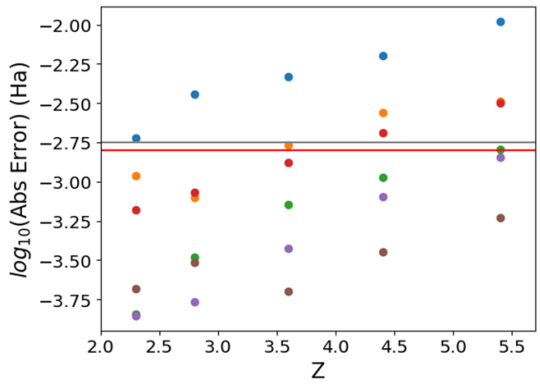


Figure 5. MAE of eigenvalues predicted by the PINN model on a logarithmic scale for the one-dimensional atom. The MAE is averaged over 10 network initializations. The red horizontal line denotes chemical accuracy, while the gray horizontal line depicts the MAE averaged over all test systems. The colors label the eigenstates in increasing order as follows: blue, orange, red, green, purple, and brown.

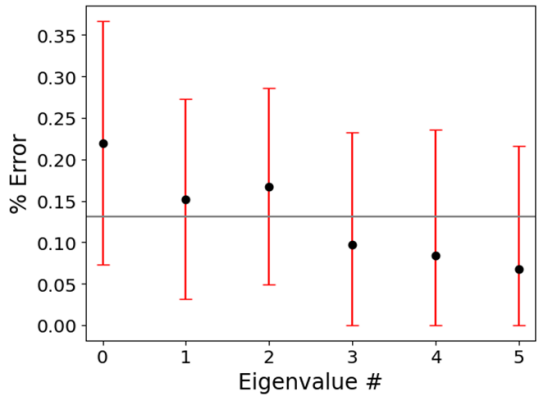


Figure 6. The MAPE resolved for the first six eigenvalues of the one-dimensional atom with varying charges predicted by the PINN model. The red error bars denote three standard deviations away from the mean, while the gray horizontal line depicts the MAPE averaged over all eigenvalues.

3.1.2. Diatomic molecule with varying charges and bond length

When switching to the diatomic molecule, the learning task becomes much more complex, as both the charges (Z_1 and Z_2) and the bond length (d) are variable parameters. This is exemplified in the roughly 10-fold increase in errors within the predicted eigenvalues. The MAE is shown in figure 7. Compared to the atomic test system, which had an overall MAE averaging around 1.77×10^{-3} Ha, the overall MAE has increased to 1.25×10^{-2} Ha (refer to the gray horizontal line in figure 7). In the atomic test system, the largest errors were observed in the ground, first, and second states of the KS spectrum. All eigenvalues exhibit some errors above the limit of chemical accuracy. This suggests that the PINN model has difficulty predicting the tails of the XC potential, but not to the same extent as placing and accurately estimating the well depth. The relative ordering of errors on the states remains consistent.

We have further broken down the error analysis into individual eigenvalues in figure 8. We display the MAPE for the first six eigenvalues, which exhibits the larger errors for the ground and first excited states observed in the MAEs plot. The overall MAPE averages around 0.575% (denoted by a gray horizontal line in figure 8).

3.1.3. Extrapolation for diatomic molecule

Purely data-driven neural network approaches rapidly degrade in predictive performance when provided with data outside the training set. However, using physics-based approaches such as PINNs and also FNOs, higher extrapolation performance can be expected. Compared to data-driven learning, the neural network in the PINN model represents the solution to a given PDE, allowing accurate predictions within the specific scope of that PDE. We evaluate the PINN model's extrapolation ability by asking it to predict on a set of densities outside the training set.

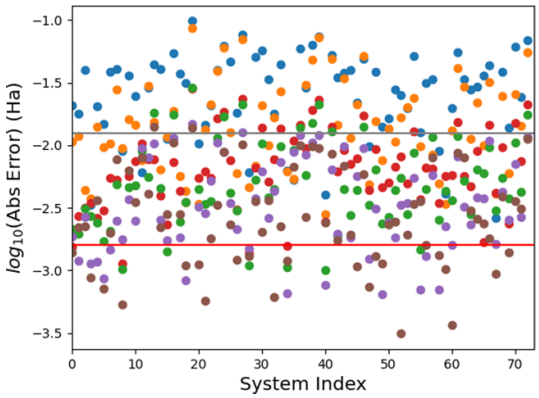


Figure 7. MAE of eigenvalues predicted by the PINN model on a logarithmic scale for the one-dimensional diatomic molecule. The MAE is averaged over 10 network initializations. The red horizontal line denotes chemical accuracy, while the gray horizontal line depicts the MAE averaged over all test systems. The colors label the eigenstates in increasing order of energy as described in figure 5.

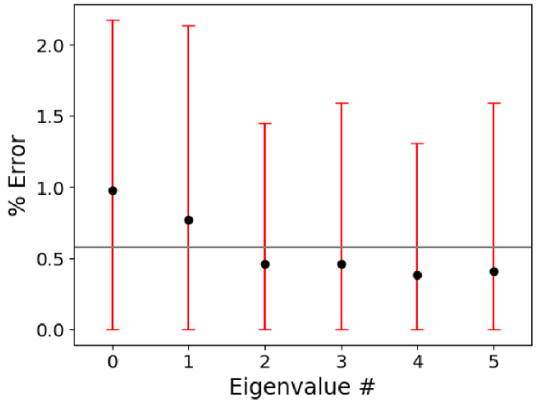


Figure 8. The MAPE resolved for the first six eigenvalues of the one-dimensional diatomic molecule with varying charges and bond length predicted by the PINN model. The red error bars denote three standard deviations away from the mean, while the gray horizontal line depicts the MAPE averaged over all eigenvalues.

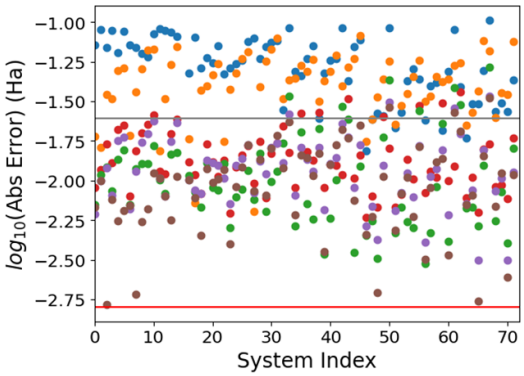


Figure 9. Extrapolation with the PINN model for the one-dimensional diatomic molecule with varying charges and bond length. The test set includes molecules with charges and bond length that are outside the parameter ranges of the training dataset. The MAE of the predicted eigenvalues is shown on a logarithmic scale. The MAE is averaged over 10 network initializations. The red horizontal line denotes chemical accuracy, while the gray horizontal line depicts the MAE averaged over all test systems. The colors labels the eigenstates in increasing order of energy as described in figure 5.

The outcome of this analysis is illustrated in figure 9, which shows the MAE. The errors have increased as anticipated, resulting in an overall MAE of 2.48×10^{-2} Ha indicated by the gray horizontal line. The error has doubled in comparison to the interpolation task featured in figure 5. While this is not a catastrophic failure, the higher error suggests the intricacy of extrapolation. The challenge of extrapolation is evident in figure 10, where we illustrate the MAPE resolved over the first six eigenvalues with an overall MAPE of

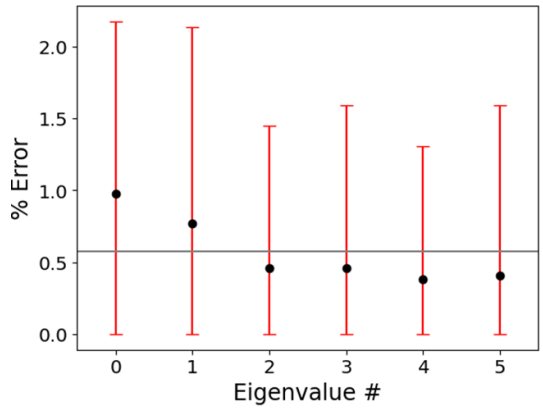


Figure 10. Extrapolation with the PINN model for the one-dimensional diatomic molecule. The MAPE is resolved for the first six eigenvalues. The red error bars denote three standard deviations away from the mean, while the gray horizontal line depicts the MAPE averaged over all eigenvalues.

0.73%, denoted by a gray line. Although no significant or sudden rise in error is observed with increasing eigenvalues, the overall MAPE is noticeably higher, as is the standard deviation shown by the red error bars.

3.2. Data-driven inversions with FNOs

In contrast to the PINN model, the density-to-potential mapping data is explicitly used in the loss function to train the data-driven FNOs, as outlined in section 2.4. The network takes the density grid as input and produces the corresponding potential grid as output. FNOs have several benefits, including learning the mapping in the functional space, which enables them to generalize effectively over varying input densities. Additionally, the network is resolution-invariant, allowing it to assess higher resolution grids than those present in the training dataset.

3.2.1. Atom with varying charge

In the case of the one-dimensional atom with varying charge Z, the FNO model demonstrates exceptional performance over the range of charge values. The error plots for the eigenvalues, as shown in figure 11, indicate that the predicted eigenvalues closely align with the true values. We resolve the error analysis over the spectrum of eigenvalues by showing the MAPE for the first six eigenvalues in figure 12. The red error bars denote three standard deviations away from the mean, demonstrating that the MAPE is within 0.15% in this range of eigenvalues. The range of MAPEs is lowest for the ground state and shows a slight increasing trend with higher energy levels. This increase can be attributed to the magnitude of the eigenvalues decreasing as the energy level increase. Higher energy states, with their inherently lower magnitudes, offer a smaller margin for absolute error, thus demanding greater accuracy. The model achieves this to a lesser degree, although the level of error is still well within the range of chemical accuracy. The overall MAE averaged over all model systems is 2.22×10^{-4} Ha and the corresponding averaged MAPE over all eigenvalues is $3.18 \times 10^{-2}\%$ denoted by gray horizontal lines in figures 11 and 12. This underscores the FNO model's ability to accurately capture the potential-density mapping.

3.2.2. Diatomic molecule with varying charges and bond length

After demonstrating improved performance over the PINNs model for the one-dimensional atom, we investigate the FNO model's abilities for the one-dimensional diatomic molecule. Variations in both charges (Z_1 and Z_2) and bond length d pose a different set of challenges for the FNO model. As depicted in figure 13, the plots for eigenvalue errors indicate a slightly higher margin of error in comparison to the atomic model system. The eigenvalue errors consistently remain within chemical accuracy, except for a particular outlier. Figure 14 shows the MAPE range for each eigenvalue. There is a clear trend of increasing MAPE range from the ground state to excited states. Nevertheless, the overall MAE of 2.27×10^{-4} Ha and MAPE of $1.90 \times 10^{-2}\%$ (denoted by gray horizontal lines) align with the results of the atomic model system, confirming the FNO model's adaptability and accuracy across this more challenging set of system parameters.

3.2.3. Extrapolation for diatomic molecule

The true test of a predictive model lies in its ability to generalize beyond the parameters of the systems in the training dataset. Figure 15 showcases the eigenvalue error plots for this dataset, which include systems with

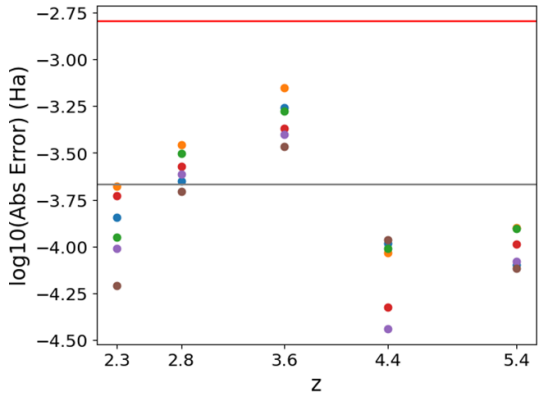


Figure 11. MAE of eigenvalues predicted by the FNO model on a logarithmic scale for the one-dimensional atom. The MAE is averaged over 10 network initializations. The red horizontal line denotes chemical accuracy, while the gray horizontal line depicts the MAE averaged over all test systems. The colors label the eigenstates in increasing order of energy as described in figure 5.

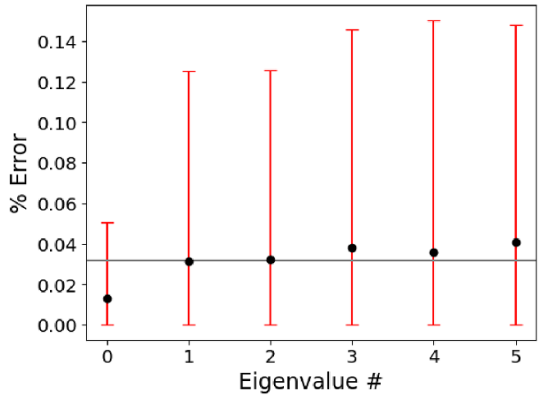


Figure 12. The MAPE resolved for the first six eigenvalues of the one-dimensional atom with varying charge predicted by the FNO model. The red error bars denote three standard deviations away from the mean, while the gray horizontal line depicts the MAPE averaged over all eigenvalues.

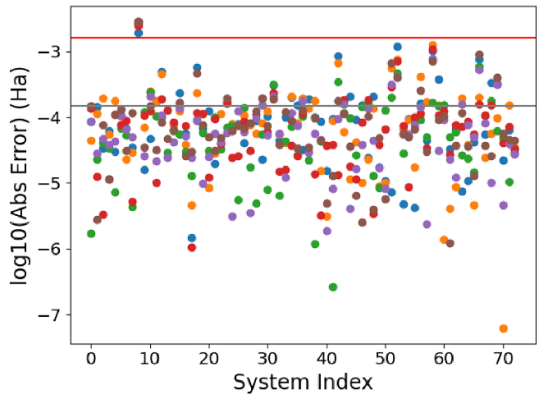


Figure 13. MAE of eigenvalues predicted by the FNO model on a logarithmic scale for the one-dimensional diatomic molecule. The MAE is averaged over 10 network initializations. The red horizontal line denotes chemical accuracy, while the gray horizontal line depicts the MAE averaged over all test systems. The colors label the eigenstates in increasing order of energy, as described in figure 5.

charges (Z_1 and Z_2) and bond lengths (d) that are outside the previous datasets. The errors recorded were marginally higher than those for the training dataset, and about half the eigenvalues are predicted within chemical accuracy. The percentage errors are within 2%, as depicted in figure 16. The FNO model yields overall an MAE of 5.4×10^{-3} Ha and MAPE of 0.25% (denoted by gray horizontal lines). This demonstrates the FNO model's robustness and potential for broader applicability in solving the KS inversion problem.

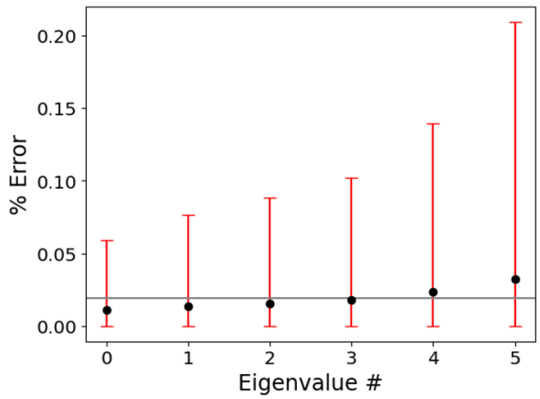


Figure 14. The MAPE resolved for the first six eigenvalues of the one-dimensional diatomic molecule with varying charges and bond length predicted by the FNO model. The red error bars denote three standard deviations away from the mean, while the gray horizontal line depicts the MAPE averaged over all eigenvalues.

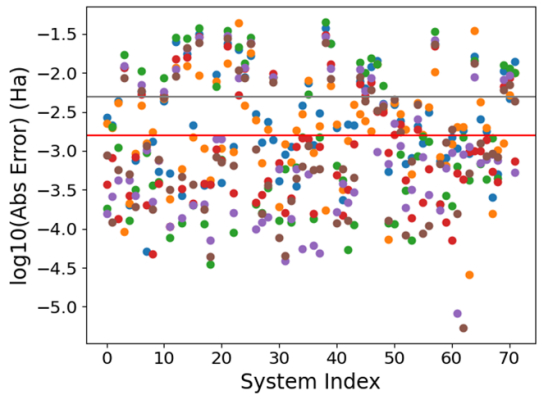


Figure 15. Extrapolation with the FNO model for the one-dimensional diatomic molecule. The test set includes molecules with charges and bond length that are outside the parameter ranges of the training dataset. The MAE of the predicted eigenvalues is shown on a logarithmic scale. The MAE is averaged over 10 network initializations. The red horizontal line denotes chemical accuracy, while the gray horizontal line depicts the MAE averaged over all test systems. The colors labels the eigenstates in increasing order of energy as described in figure 5.

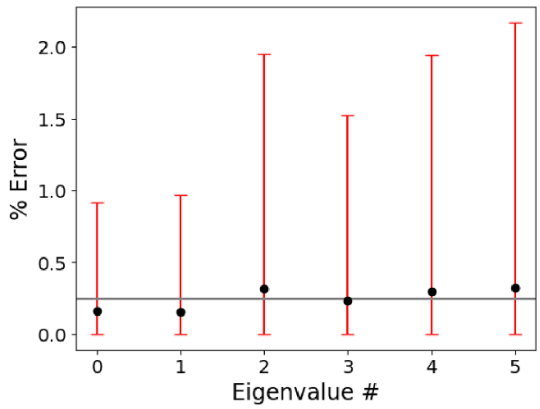


Figure 16. Extrapolation with the FNO model for the one-dimensional diatomic molecule. The MAPE is resolved for the first six eigenvalues. The red error bars denote three standard deviations away from the mean, while the gray horizontal line depicts the MAPE averaged over all eigenvalues.

4. Discussion

We summarize and contrast our results for the PINN and FNO models in table 1 for both atomic and molecular model systems. Our performance metrics are the MAE expressed in Hartree (Ha) and the MAPE.

Table 1. Performance comparison of PINN and FNO models on both model systems, showing the MAE (in Ha) and MAPE (%) and denoting the maximum absolute errors and the maximum percentage errors in braces. The labels 301 and 501 indicate the grid resolutions and E denotes extrapolation.

Model	Atom (301)	Atom (501)	Molecule (301)	Molecule (301 E)
PINN	0.0018 Ha *(0.0110 Ha)* 0.13% *(0.30 %)*	0.0007 Ha *(0.0034 Ha)* 0.06% *(0.15%)*	0.0125 Ha *(0.1000 Ha)* 0.58% *(2.37%)*	0.0248 Ha *(0.1200 Ha)* 0.73% *(3.25%)*
FNO	0.0002 Ha *(0.0009 Ha)* 0.03% *(0.15%)*	0.0002 Ha *(0.0010 Ha)* 0.03% *(0.19%)*	0.0002 Ha *(0.0051 Ha)* 0.02% *(0.46%)*	0.0054 Ha *(0.0584 Ha)* 0.25% *(3.51%)*

Table 2. Computational timings for inversions (in seconds), comparing a conventional method (iDEA) with the PINN and FNO models.

Model	Atom (301)	Molecule (301)	Molecule (501)
iDEA	135 ± 29	305 ± 262	806 ± 447
PINN	0.0016 ± 0.0011	0.0026 ± 0.0013	N/A
FNO	0.0022 ± 0.0009	0.0020 ± 0.0003	0.0037 ± 0.0004

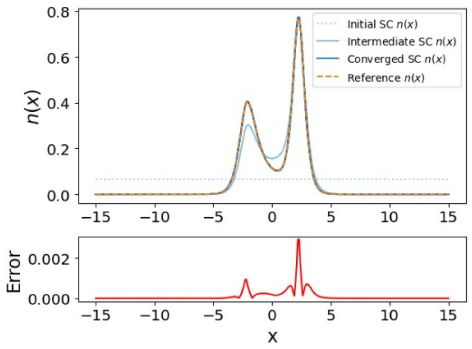
Overall, the FNO outperforms the PINN model, providing chemical accuracy (MAE $\leqslant$ 0.0016 Ha) for almost all model systems.

For atoms, the PINN model shows an MAE of 0.0018 Ha, corresponding to a relative error of 0.13%, while the FNO model shows an MAE of 0.0002 Ha, corresponding to a relative error of 0.03%. The PINN model significantly improves its accuracy when the grid resolution is increased to 501, while the FNO model shows consistent accuracy over different grid resolutions.

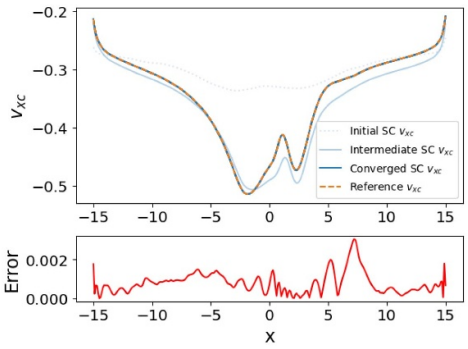
When applied to molecules, the error of the PINN model increases significantly, especially for the extrapolation task, with an MAE reaching 0.0248 Ha and a relative error up to 0.73%. The FNO model maintains a lower error rate and remains within the chemical accuracy except for the extrapolation task, where its MAE increases to 0.0054 Ha and a relative error of 0.25%. Despite the increase in error, the FNO model is still highly accurate. The values in parentheses represent additional metrics, namely the maximum absolute error and the maximum percentage error, which indicate the maximum spread of errors to evaluate the robustness of the models. Of additional note is the fact that FNOs are resolution invariant, indicated by the N/A in table 2. Here we trained the FNO on data from a set with a grid resolution of 301 points, which could be used to infer on a density generated on a grid with 501 points. The convolutional PINNs presented in this paper are not able to do this.

Having established the accuracy of the PIML methods, we turn to analyze the computational efficiency. To this end, table 2 shows the computational timings of the PINN and FNO models against a conventional inversion method (iDEA code) across the different model systems. Timing results are given in seconds, with standard deviations to indicate variability in computational performance. For atoms, the conventional inversion method shows a mean computation time of 135 ± 29 seconds, which is significantly higher than both PINN and FNO models, which show mean computation times of 0.0016 ± 0.0011 seconds and 0.0022 ± 0.0009 seconds, respectively. The difference is even more pronounced for the molecular systems, where the conventional method requires 305 ± 262 seconds, while PINN and FNO remain in the millisecond range. In particular, for the molecular system with larger grid resolution (501 grid points), the computational time of the conventional method continues to increase, while FNO maintains an efficient computational evaluation in the millisecond range. These results highlight the substantial improvements in computational speed offered by the PINN and FNO models over conventional methods, with FNO showing slightly faster times and lower variability than PINN in the scenarios tested, suggesting a robustness that could be particularly beneficial for larger and more complex systems.

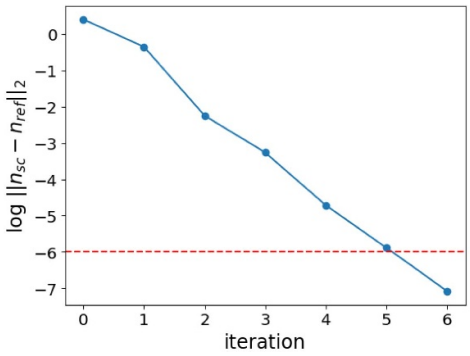
Looking beyond the main scope of this work, the PIML inversion scheme provides an alternative route to forward-direction self-consistent DFT calculations. In figure 17, our FNO models for the one-dimensional molecular system have been implemented inside of a self-consistent field cycle. This implementation exhibits rapid convergence to reference densities in six iterations, even starting from an initial density missing all the characteristic features found in the reference. Similar results are found for other example systems, shown in the Supplementary Material. This application of our results illustrates a pragmatic approach to using this or other reliable inversion methods in practical calculations.



(a) Density convergence through self-consistent (SC) calculations.



(b) v_{xc} convergence through self-consistent (SC) calculations.



(c) Log norm through the SC calculation. The norm is $\|n_{sc} - n_{ref}\|_2$. The red dotted line indicates the selected convergence criterion value of 10^{-6}.

Figure 17. Example of self-consistent calculation for a molecular system with $d = 4.5, z_1 = 2.0, z_2 = 4.5$. Starting with a constant initial density, the calculations are iteratively carried out using the FNO model to calculate the v_{xc} at each iteration. For (a) and (b), the dotted light blue line is the initial density/potential, solid light blue line represents intermediate stages and the solid dark blue line is the converged density/potential. The dashed orange line is the exact density/potential obtained from iDEA. The bottom plot shows the absolute difference between the converged SC quantity and reference quantity.

5. Conclusion

In summary, this work has demonstrated the potential of PIML techniques, specifically PINNs and FNOs, to address the inverse problem in KS-DFT. The PINN model uses the underlying KS differential equation to predict the KS potential from given electron densities. FNOs, on the other hand, use data from known density-to-potential mappings. They exhibit superior performance in both accuracy and computational efficiency as PINNs across a range of grid resolutions and complexity of model systems. In particular, FNOs maintain chemical accuracy in most cases, underscoring their ability to handle the density-to-potential mapping effectively. The computational times for both PIML approaches significantly outperform conventional methods, in both direct and self-consistent calculations, offering a promising alternative for large-scale electronic structure inversions.

A promising avenue for future work is to combine the strengths of both PINNs and FNOs. This could be realized by a hybrid model with a two-component loss term: one part based on the density-to-potential mapping characteristic of the FNO model, and another part derived from the underlying KS differential equation represented by the PINN model. During training, both the FNO and PINN components would be optimized simultaneously. This approach is expected to yield an efficient PIML model that not only exploits the data-driven robustness of the FNO component, but also requires less data and exhibits broader generalizability due to the PINN-derived loss. Such models, referred to as physics-informed neural operators (PINOs), have been tested on various PDE benchmarks [88, 89]. A test of the robustness of the PINO approach would be applying it to KS inversion for real systems.

Our results indicate that the integration of machine learning, in particular PIML techniques, can significantly improve the efficiency and accuracy of electronic structure inversions. These advances offer promising prospects for elucidating the XC potential of materials from experimental electron density data

and for advancing the development of XC functionals within DFT, providing a reliable computational tool for accurate density-to-potential inversions.

Data availability statement

The data that support the findings of this study are openly available at the following URL/DOI: https://rodare.hzdr.de/record/2720.

Acknowledgments

This work was partially supported by the Center for Advanced Systems Understanding (CASUS) which is financed by Germany's Federal Ministry of Education and Research (BMBF) and by the Saxon state government out of the State budget approved by the Saxon State Parliament. Computations were performed on a Bull Cluster at the Center for Information Services and High-Performance Computing (ZIH) at Technische Universität Dresden and on the cluster Hemera of the Helmholtz-Zentrum Dresden-Rossendorf (HZDR). A C and K S acknowledge funding from the Helmholtz Association's Initiative and Networking Fund through Helmholtz AI.

ORCID iDs

Vincent Martinetto https://orcid.org/0000-0001-6026-7397
Karan Shah https://orcid.org/0000-0002-5480-2880
Attila Cangi https://orcid.org/0000-0001-9162-262X
Aurora Pribram-Jones https://orcid.org/0000-0003-0244-1814

References

[1] Hohenberg P and Kohn W 1964 *Phys. Rev.* **136** B864–71
[2] Kohn W and Sham L J 1965 *Phys. Rev.* **140** A1133–8
[3] Perdew J P and Schmidt K 2001 *AIP Conf. Proc.* **577** 1–20
[4] Lee C, Yang W and Parr R G 1988 *Phys. Rev.* B **37** 785–9
[5] Perdew J P and Wang Y 1992 *Phys. Rev.* B **45** 13244–9
[6] Ceperley D M and Alder B J 1980 *Phys. Rev. Lett.* **45** 566–9
[7] Perdew J P, Burke K and Ernzerhof M 1996 *Phys. Rev. Lett.* **77** 3865–8
[8] Becke A D 1988 *Phys. Rev.* A **38** 3098–100
[9] Perdew J P, Ruzsinszky A, Csonka G I, Vydrov O A, Scuseria G E, Constantin L A, Zhou X and Burke K 2008 *Phys. Rev. Lett.* **100** 136406
[10] Tao J, Perdew J P, Staroverov V N and Scuseria G E 2003 *Phys. Rev. Lett.* **91** 146401
[11] Sun J, Ruzsinszky A and Perdew J P 2015 *Phys. Rev. Lett.* **115** 036402
[12] Becke A D 1993 *J. Chem. Phys.* **98** 5648–52
[13] Adamo C and Barone V 1999 *J. Chem. Phys.* **110** 6158–70
[14] Langreth D C and Perdew J P 1977 *Phys. Rev.* B **15** 2884–901
[15] Furche F 2001 *Phys. Rev.* B **64** 195120
[16] Fuchs M and Gonze X 2002 *Phys. Rev.* B **65** 235109
[17] Harl J and Kresse G 2008 *Phys. Rev.* B **77** 045136
[18] Furche F and Van Voorhis T 2005 *J. Chem. Phys.* **122** 164106
[19] Chen G P, Voora V K, Agee M M, Balasubramani S G and Furche F 2017 *Annu. Rev. Phys. Chem.* **68** 421–45
[20] Grimme S 2006 *J. Chem. Phys.* **124** 034108
[21] Tarnopolsky A, Karton A, Sertchook R, Vuzman D and Martin J M L 2008 *J. Phys. Chem.* A **112** 3–8
[22] Karton A, Tarnopolsky A, Lamére J F, Schatz G C and Martin J M L 2008 *J. Phys. Chem.* A **112** 12868–86
[23] Martin J M L and Santra G 2020 *Isr. J. Chem.* **60** 787–804
[24] Pribram-Jones A, Gross D A and Burke K 2015 *Annu. Rev. Phys. Chem.* **66** 283–304
[25] Medvedev M G, Bushmarinov I S, Sun J, Perdew J P and Lyssenko K A 2017 *Science* **355** 49–52
[26] Fiedler L, Shah K, Bussmann M and Cangi A 2022 *Phys. Rev. Mater.* **6** 040301
[27] Pederson R, Kalita B and Burke K 2022 *Nat. Rev. Phys.* **4** 357–8
[28] Liu Q, Wang J, Du P, Hu L, Zheng X and Chen G 2017 *J. Phys. Chem.* A **121** 7273–81
[29] Lei X and Medford A J 2019 *Phys. Rev. Mater.* **3** 063801
[30] Chen Y, Zhang L, Wang H E W 2021 DeePKS-kit: a package for developing machine learning-based chemically accurate energy and density functional models (arXiv:2012.14615)
[31] Margraf J T and Reuter K 2021 *Nat. Commun.* **12** 344
[32] Nelson J, Tiwari R and Sanvito S 2019 *Phys. Rev.* B **99** 075132
[33] Schmidt J, Benavides-Riveros C L and Marques M A L 2019 *J. Phys. Chem. Lett.* **10** 6425–31
[34] Suzuki Y, Nagai R and Haruyama J 2020 *Phys. Rev.* A **101** 050501
[35] Yu M, Yang S, Wu C and Marom N 2020 *npj Comput. Mater.* **6** 1–6
[36] Bogojeski M, Vogt-Maranto L, Tuckerman M E, Müller K R and Burke K 2020 *Nat. Commun.* **11** 5223
[37] Griego C D, Zhao L, Saravanan K and Keith J A 2020 *AIChE J.* **66** e17041
[38] Dick S and Fernandez-Serra M 2020 *Nat. Commun.* **11** 3509

[39] Fujinami M, Kageyama R, Seino J, Ikabata Y and Nakai H 2020 *Chem. Phys. Lett.* **748** 137358
[40] Nagai R, Akashi R and Sugino O 2020 *npj Comput. Mater.* **6** 1–8
[41] Mezei P D and von Lilienfeld O A 2020 *J. Chem. Theory Comput.* **16** 2647–53
[42] Lentz L C and Kolpak A M 2020 *J. Phys.: Condens. Matter* **32** 155901
[43] Sun G and Sautet P 2019 *J. Chem. Theory Comput.* **15** 5614–27
[44] Zhu J, Vuong V Q, Sumpter B G and Irle S 2019 *MRS Commun.* **9** 867–73
[45] Peccati F, Desmedt E and Contreras-García J 2019 *Comput. Theor. Chem.* **1159** 23–26
[46] Hollingsworth J, Li L, Baker T E and Burke K 2018 *J. Chem. Phys.* **148** 241743
[47] Vegge T, Christensen R, Dickens C and Hansen H 2018 *ACS National Meeting Book of Abstracts* vol 255
[48] Messina L, Quaglino A, Goryaeva A, Marinica M C, Domain C, Castin N, Bonny G and Krause R 2020 *Nucl. Instrum. Methods Phys. Res.* B **483** 15–21
[49] Nagai R, Akashi R and Sugino O 2022 *Phys. Rev. Res.* **4** 013106
[50] Bystrom K and Kozinsky B 2022 *J. Chem. Theory Comput.* **18** 2180–92
[51] Kalita B, Pederson R, Chen J, Li L and Burke K 2022 *J. Phys. Chem. Lett.* **13** 2540–7
[52] Sabatier P C 2000 *J. Math. Phys.* **41** 4082–124
[53] Kirsch A 2022 *An Introduction to the Mathematical Theory of Inverse Problems* (Springer International Publishing)
[54] Wang Y and Parr R G 1993 *Phys. Rev.* A **47** R1591–3
[55] Jensen D and Wasserman A 2018 *Int. J. Quantum Chem.* **118** e25425
[56] Callow T J, Lathiotakis N N and Gidopoulos N I 2020 *J. Chem. Phys.* **152** 164114
[57] Zhao Q, Morrison R C and Parr R G 1994 *Phys. Rev.* A **50** 2138–42
[58] Wu Q and Yang W 2003 *J. Chem. Phys.* **118** 2498–509
[59] Wagner L O, Stoudenmire E M, Burke K and White S R 2012 *Phys. Chem. Chem. Phys.* **14** 8581–90
[60] van Leeuwen R and Baerends E J 1994 *Phys. Rev.* A **49** 2421–31
[61] Yang Z-H, Trail J R, Pribram-Jones A, Burke K, Needs R J and Ullrich C A 2014 *Phys. Rev.* A **90** 042501
[62] Eich F G and Hellgren M 2014 *J. Chem. Phys.* **141** 224107
[63] Luo K, Elliott P and Maitra N T 2013 *Phys. Rev.* A **88** 042508
[64] Elliott P, Fuks J I, Rubio A and Maitra N T 2012 *Phys. Rev. Lett.* **109** 266404
[65] Dar D, Lacombe L and Maitra N T 2022 *Chem. Phys. Rev.* **3** 031307
[66] Gedeon J, Schmidt J, Hodgson M J P, Wetherell J, Benavides-Riveros C L and Marques M A L 2021 *Mach. Learn.: Sci. Technol.* **3** 015011
[67] Cohen A J, Mori-Sánchez P and Yang W 2008 *Science* **321** 792–4
[68] Wang F, Eljarrat A, Müller J, Henninen T R, Erni R and Koch C T 2020 *Sci. Rep.* **10** 5730
[69] Genzel M, Macdonald J and März M 2023 *IEEE Trans. Pattern Anal. Mach. Intell.* **45** 1119–34
[70] Karniadakis G E, Kevrekidis I G, Lu L, Perdikaris P, Wang S and Yang L 2021 *Nat. Rev. Phys.* **3** 422–40
[71] Raissi M, Perdikaris P and Karniadakis G 2019 *J. Comput. Phys.* **378** 686–707
[72] Li Z, Kovachki N B, Azizzadenesheli K, Liu B and Bhattacharya K Stuart A and Anandkumar A 2020 Fourier neural operator for parametric partial differential equations *Int. Conf. on Learning Representations*
[73] Yang L, Meng X and Karniadakis G E 2021 *J. Comput. Phys.* **425** 109913
[74] Chen Y, Lu L, Karniadakis G E and Negro L D 2020 *Opt. Express* **28** 11618–33
[75] Lou Q, Meng X and Karniadakis G E 2021 *J. Comput. Phys.* **447** 110676
[76] Kovachki N, Li Z, Liu B, Azizzadenesheli K, Bhattacharya K, Stuart A and Anandkumar A 2023 Neural operator: learning maps between function spaces (arXiv:2108.08481)
[77] Englert B G 1988 *Semiclassical Theory of Atoms* (Springer)
[78] Hodgson M J P, Ramsden J D, Chapman J B J, Lillystone P and Godby R W 2013 *Phys. Rev.* B **88** 241102
[79] Adamson S *et al* 2023 iDEA (interacting Dynamic Electrons Approach) iDEA Software Library (available at: https://github.com/iDEA-org/iDEA#readme)
[80] Martinetto V, Shah K, Cangi A and Pribram-Jones A 2024 (available at: https://rodare.hzdr.de/record/2720)
[81] Krizhevsky A, Sutskever I and Hinton G E 2017 *Commun. ACM* **60** 84–90
[82] Simonyan K and Zisserman A 2015 Very deep convolutional networks for large-scale image recognition (arXiv:1409.1556)
[83] He K, Zhang X, Ren S and Sun J 2016 Deep residual learning for image recognition *2016 IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)* pp 770–8
[84] Szegedy C, Liu W, Jia Y, Sermanet P, Reed S, Anguelov D, Erhan D, Vanhoucke V and Rabinovich A 2015 Going deeper with convolutions *2015 IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)* pp 1–9
[85] Lu L, Jin P and Karniadakis G E 2019 CoRR (arXiv:1910.03193)
[86] Anandkumar A, Azizzadenesheli K, Bhattacharya K, Kovachki N, Li Z, Liu B and Stuart A 2020 Neural operator: graph kernel network for partial differential equations *ICLR 2020 Workshop on Integration of Deep Neural Models and Differential Equations*
[87] Li Z, Kovachki N, Azizzadenesheli K, Liu B and Bhattacharya K Stuart A and Anandkumar A 2020 Multipole graph neural operator for parametric partial differential equations *Proc. 34th Int. Conf. on Neural Information Processing Systems NIPS'20* (Curran Associates Inc.) pp 6755–66
[88] Wang S, Wang H and Perdikaris P 2021 *Sci. Adv.* **7** eabi8605
[89] Li Z, Zheng H, Kovachki N, Jin D, Chen H, Liu B, Azizzadenesheli K and Anandkumar A 2023 Physics-informed neural operator for learning partial differential equations (arXiv:2111.03794 [cs, math])