scientific reports

Check for updates

OPEN

Physics informed neural network for charged particles surrounded by conductive boundaries

Fatemeh Hafezianzade[1], Morad Biagooi[2] & SeyedEhsan Nedaaee Oskoee[1,3]✉

Molecular dynamics of charged particles in porous conductive media have received considerable attention in recent years due to their application in cutting-edge technologies such as batteries and supercapacitors. Due to the presence of long-range electrical interactions, induced charges present at the boundary, and the influence of boundary conditions, the simulation of these systems is more challenging than the simulation of typical molecular dynamic systems. Simulating these kinds of systems typically involves using a numerical solver to solve the Poisson equation, which is a very time-consuming procedure. Recently, Physics-Informed Neural Networks (PINNs) have been introduced as an alternative to numerical solutions of PDEs. In this paper, we present a new PINN-based model for predicting the potential of point-charged particles surrounded by conductive walls. As a result of the proposed PINN model, the mean square error is less than $\prime$ and $_{\kappa}$ score is more than 90% for the corresponding example simulation. Results have been compared with typical neural networks and random forest as standard machine learning algorithms. The R^2 score of the random forest model was 70%, and a standard neural network could not be trained well.

Computational Electromagnetic Simulation plays a significant role in many areas of science and engineering, such as soft matter, electrical engineering, biomedical engineering and chemistry. In addition, it has numerous applications in industry. For example, it is one of the main tools in investigating and designing the process of supercapacitors, which are porous energy storage devices with many applications in industry, especially when high power consumption or transfer is needed[1]. Here, studying the physical mechanisms arising from charge storage in supercapacitors is essential for further technological development[2,3].

Solving Maxwell's equation, especially the Poisson equation in this study, is an essential part of computational electromagnetic algorithms[4]. Solving the Poisson equation can help scientists to calculate the potential of electrical sources in any system. However, many difficulties arise in practice due to the long-range nature of electrical interactions. In particular, estimating the potential of point-charged components in an environment with conductive walls is challenging because of the induced charges presented on the boundaries.

Generally, there are two approaches to solving the Poisson equation: analytical solution[4] and numerical methods. There are limited techniques for solving analytically, like image charges methods applicable for cases with regular geometries; however, there is no guarantee to achieve practical results. If, for example, a particle is placed in a cubic conductive container, the image charges method will produce an infinite series. On the other hand, numerical methods lead to approximate solutions based on discretizing space and/or time domains. One of the typical numerical methods is the Finite Element Method (FEM)[5], which discretizes the continuous partial differential equations (PDEs) and forms a linear set of algebraic equations[6]. Nevertheless, even FEM fails in calculating the potential in a charged particle's position since the electrical potential is singular at the place of charges. There are a number of methods and algorithms that have been developed to address this problem, including Induced Charge MMM2D (ICMMM2D)[7] for 2D, ELCIC[8] for 2D + h, Induced Charge Computation ($ICC*$)[9–11] for 3D periodicity, and a method introduced by Reed et al.[12] have been developed. In addition, recently, there has been another algorithm named PLT. It was first demonstrated for a partially periodic system constrained between two metallic plates in[13], and then it was applied to CAVIAR[14], a molecular dynamics simulation package for charged particles surrounded by non-trivial conductive boundaries. Numerical solving of these problems with the CAVIAR package is accurate; moreover, it took less time than $ICC*$[14] but is still time and memory-consuming.

[1]Department of Physics, Institute for Advanced Studies in Basic Sciences, Zanjan 45137-66731, Iran. [2]Intelligent Data Aim Ltd (IDA Ltd), Science and Technology Park of Institute for Advanced Studies in Basic Sciences, Zanjan 45137-65697, Iran. [3]Research Center for Basic Sciences and Modern Technologies (RBST), Institute for Advanced Studies in Basic Sciences (IASBS), Zanjan 45137-66731, Iran. ✉email: nedaaee@iasbs.ac.ir

Recently another data-driven approach to solving the PDEs based on deep machine learning is also of great current interest. For instance, Shan et al.[15] present a CNN to predict the electric potential with different excitations and permittivity distribution in 2D and 3D models. It is fast and efficient compared with FEM[5]. However, a couple of problems prevent it from utilizing as a Poisson solver in the MD simulation process; first, it could not work in the case of discrete density functions such as those of point charges, and second, it is a physics-free approach which makes it hard to consider boundary conditions. To overcome the first problem, one can use the PLT algorithm. Additionally, Raissi et al. introduced the physics-informed neural network (PINN) that the loss function defined by[16] is an excellent alternative to the conventional deep learning method because of the governing equations, boundary conditions and initial conditions used in its definition.

In this paper, we applied a new PINN-based model to predict the potential of point-charged particles surrounded by conductive walls. We then compared the results with typical neural networks and random forests as a standard machine learning algorithm. For instance, we tried to implement these models for a charged particle in a spherical container. The reason for utilizing this simple example was that there is an exact solution to this problem through the analytical method, the image charges method. As a starting point, we used the PLT algorithm to transfer the Poisson equation into the Laplace equation with modified boundary conditions. Then we trained the model to solve the Laplacian equation with new boundary conditions. The input data is included the position in which we want to evaluate the potential on it and the modified boundary conditions; the output data is the corresponding electrical potential of that position.

Results

In this paper, we predict the smooth potential of a point-charged particle in a spherical conductive container. First, we set the train and test set with 5000 and 1000 samples; then we train our models to predict smooth potential. We can calculate total potential by summating smooth and singular potential (more detailed in the "Methods" section). However, in this work, to compare our results with CAVIAR[14], we investigate the smooth potential.

Random forest. We optimize over the only hyperparameter, the number of trees in the forest that influences the fitting of the random forest model. In Fig. 1, we plot *MSE* (the left panel) and R^2 score (the right one) as a function of the number of trees for the test set to determine the optimal hyperparameter, which we find 100 trees since progress after 100 trees is negligible. Afterward, we trained the RF model using 100 trees.

RF prediction. Figure 2 is illustrated the RF model with 100 trees. It shows that prediction is acceptable when the numeric value of potential is less than 0.3 ($\phi_{True} < 0.3$) while it could not predict precisely in the case of $\phi_{True} >= 0.3$. The graphs of Fig. 2 compare the true potential ϕ_{True} and RF model predicted potential ϕ_{RF} for the train data set(left picture) and test data set(right image). The RF method is relatively fast; however, it works when the predicted potential is smooth and relatively small; it is not suitable in the case of point-charged particles (where a gradient of potential as well as its numeric value is high at the position of the charge). Furthermore, it fails to predict the potential near the boundaries since the gradient of the potential is considerable.

PINN based model and NN. By setting both λ_2 and λ_3 in the Eq. (9) to zero in the PINN-based model, one can get exactly the NN model. Therefore we investigate models together and report their results simultaneously in the following section.

Unlike the RF model, we define several hyperparameters for analyzing the performance of PINN and NN: a number of neurons, a number of layers, λ_2, and λ_3. To tune all hyperparameters, we train the model up to 100,000 epochs, using the L-BFGS-B optimizer[25], until the model's tolerance reaches the level of machine epsilon. We use a tanh activation function for all layers except the last one. Supplementary Table 1

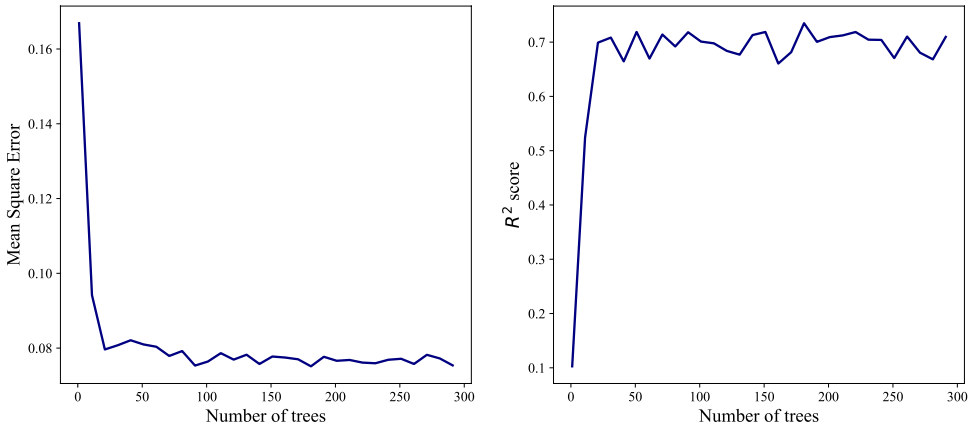

Figure 1. *MSE* (left), and R^2 score (right) for 1000 different sample of Test set.

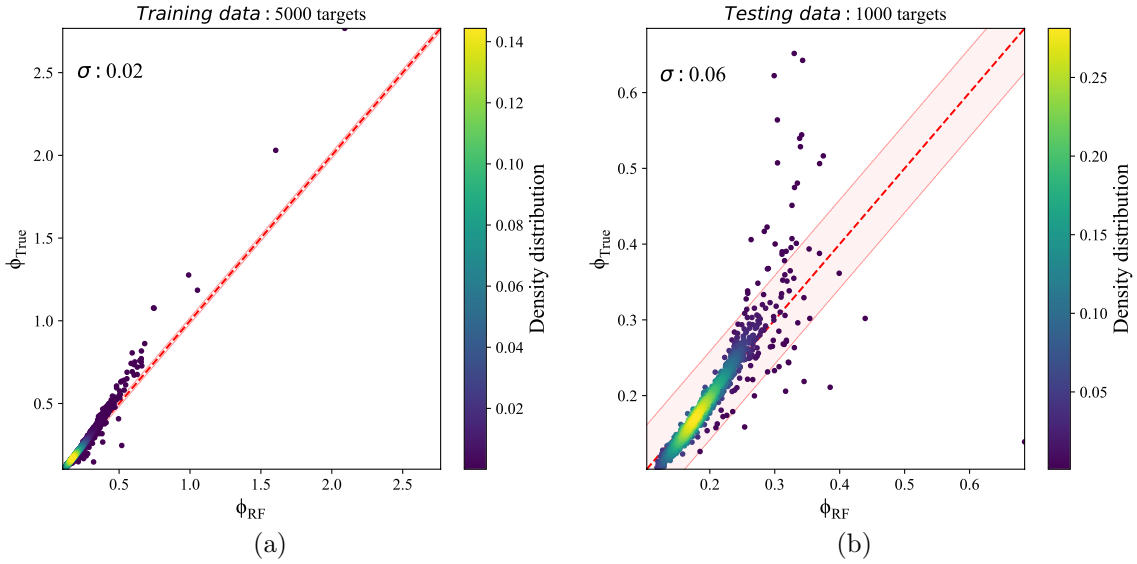

Figure 2. Potential estimation of RF model: (**a**) show the train data set with 5000 samples with scatter of $\sigma = 0.02$, (**b**) show the test data set with 1000 samples with scatter of $\sigma = 0.07$. The dashed red line shows where the predicted potential equals the true potential. The pink-shaded region marks 1σ scatter of potential errors.

is reported the *MSE* between the predicted and the same potential for a different value of hyperparameters; $\lambda_2 = [0, 0.1, 0.2, 0.3]$, $\lambda_3 = [0, 0.1, 0.2, 0.3, 0.4]$, number of hidden layers= $[1, 3, 5, 7]$ and number of neurons per hidden layer= $[10, 30, 50]$ for 1000 samples of the test set. We chose $\lambda_4 = 0.0001$ to prevent over-fitting.

PINN and NN prediction. The model with seven layers and 50 neurons per layer resulted better when λ_2 and λ_3 are 0.3, 0.3, or 0.2, 0.4, respectively (more detailed in the Supplementary Table 2). When λ_2 and λ_3 are zero, a standard neural network, the model has not worked well; it is observed from Supplementary Table 2 and Fig. 3.

Both plots in Fig. 3 compare the true and predicted potentials for the best-tuned NN and the best-tuned PINN model with 7 layers and 50 neurons per layer, $\lambda_2 = 0.3$ and $\lambda_3 = 0.3$- on the train set. As can be seen, the NN model is not trained well, while the PINN-based model could predict the potential precisely with a scatter of 0.01. Although the PINN-based model predicts the train set well, aiming to clarify that over-fitting has not been accrued, we also evaluate the model on the test set, Fig. 4.

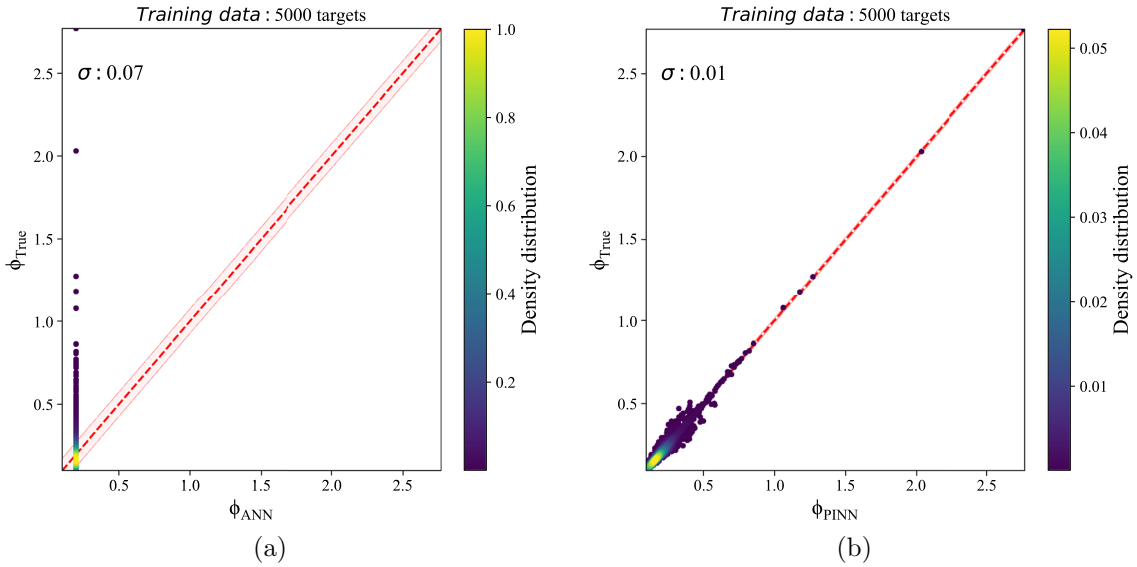

Figure 3. Potential estimation of best-tuned (**a**) NN with scatter of $\sigma = 0.07$ and (**b**) PINN model on the 5000 samples of the train set with scatter of $\sigma = 0.01$. The dashed red line shows where the predicted potential equals the true potential. The pink-shaded region marks 1σ scatter of potential errors.

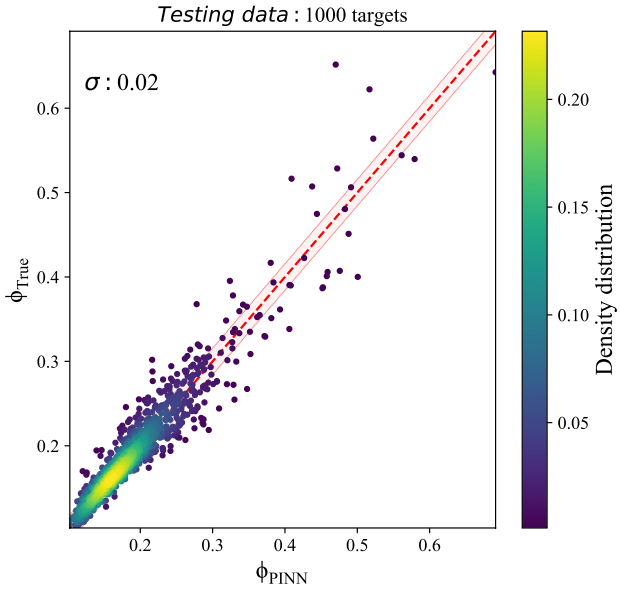

Figure 4. Potential estimation of best-tuned PINN model on the 1000 samples of the test set with scatter of $\sigma = 0.02$, $MSE = 0.069$ and $R^2 score = 0.851$. The dashed red line shows where the predicted potential equals the true potential. The pink-shaded region marks 1σ scatter of potential errors.

Comparison. We evaluate RF, NN, and PINN to estimate the potential of point-charged particles surrounded by conductive walls. According to Fig. 3, NN was not trained well, while RF and PINN-based models could predict potential precisely. However, RF did not work well to estimate $\phi_{True} > 0.3$. Apart from this, the best model could estimate not only the potential of the train and the test sets but also the potential of point-charged particles that are not in the train or test set. So we evaluate the best tuned-PINN model and RF on the extrapolation samples; the results are reported in Supplementary Table 3. As can be seen in Fig. 5 PINN-based model could predict the potential of newly charged particles better than the RF model, where PINN could predict $\phi_{True} > 0.3$ by far better than RF.

Generalization (multi charged particles). For generalization, we test the PINN-based model with $\lambda_2 = 0.3$ and $\lambda_3 = 0.3$ for the case of more than one charged particle surrounded with conductive boundaries. Since the Laplace equation is a linear function, we predict the potential of each charged particle and then

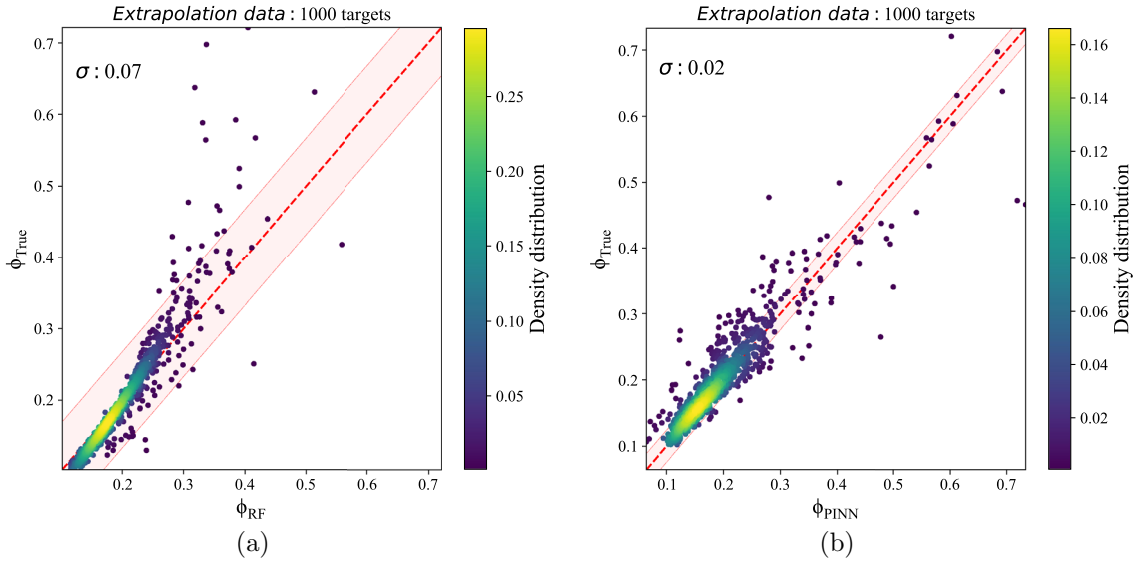

Figure 5. Potential estimation of best-tuned (**a**) RF with scatter of $\sigma = 0.07$, and (**b**) PINN-based model with scatter of $\sigma = 0.02$ on the 1000 samples of the Extrapolation set. The dashed red line shows where the predicted potential equals the true potential. The pink-shaded region marks 1σ scatter of potential errors.

calculate the total potential with a superposition of the corresponding predicted potential. After that, we report the *MSE* between the predicted smooth potential and the exact smooth solution, which is calculated by the image charges method. Figure 6 shows the relation between *MSE* and *N*, the number of charged particles. As expected, the *MSE* is independent of the number of charged particles. Therefore, it leads to the fact that we can also use this method for problems with any desired particles. In an accurate MD simulation, the number of particles is significantly higher than shown in Fig. 6. Since there is no notable trend between the *MSE* and the number of particles *N*, one can conclude that PINN's prediction is valid for any number of particles. It is also applicable to complex cases such as charged macromolecules and polymers. Recent simulations have shown that the superposition principle works well for complex systems using the PLT algorithm[26]. Therefore, PINN-based methods are well suited for these complex components.

Conclusion

In this study, we have trained a machine to predict the smooth potential of charged components surrounded by conductive boundaries. In this scene, the total potential could be easily calculated by the summation of predicted smooth potential with singular potential due to the PLT algorithm. The reference set consists of analytic solutions, the solution of the image charge method, which is split into a train set with 5000 samples and a test set with 1000 samples. To check the accuracy of our model, we set another data set called the extrapolation set consisting of 1000 samples with different boundary conditions that were not in the train or even the test set. Our main conclusion can be summarized as follows:

- We find that the PINN-based model trained better than RF and NN models. RF could not predict high potential; on the other hand, the NN could not be trained well at all.
- our PINN-based model could predict the potential of the test set with $MSE = 0.069$, $R^2 score = 0.902$, and scatter $\sigma = 0.02$. It also could predict the potential of the extrapolation set with $MSE = 0.089$, $R^2 score = 0.851$, and scatter $\sigma = 0.02$.
- Since the Laplace equation is a linear equation, the trained model could predict the potential of more than one charged particle by summating every particle's predicted potential. Besides, we show that the *MSE* of more than one particle is independent of a number of particles.

We used an analytical approach, the image charge method, to solve the Laplace equation in order to construct the test and training sets. Analytical methods work well for simple geometries, like conductive spherical shells in our case. In contrast, most simulations contain complex geometries that require numerical methods, such as the Finite Element method, in order to solve the Laplace equation. These methods are usually time-consuming, reducing simulation efficiency and becoming an obstacle for simulating large systems with numerous particles. A trained PINN model, on the other hand, is much more efficient; it is as simple as multiplying network inputs with weight tensors and transforming the resulting numbers to predict the numerical value of the electrical potential at any given point inside the simulation box. In contrast to other numerical methods, PINN provides a powerful tool for predicting the electrical potential in any desired point and computing the electrical forces much more quickly. As a result, we are able to consider a more complex system with more particles. Consequently, we can sample more data for averaging, resulting in less finite size error. This compensates for the lack of high-precision results compared with other numerical results.

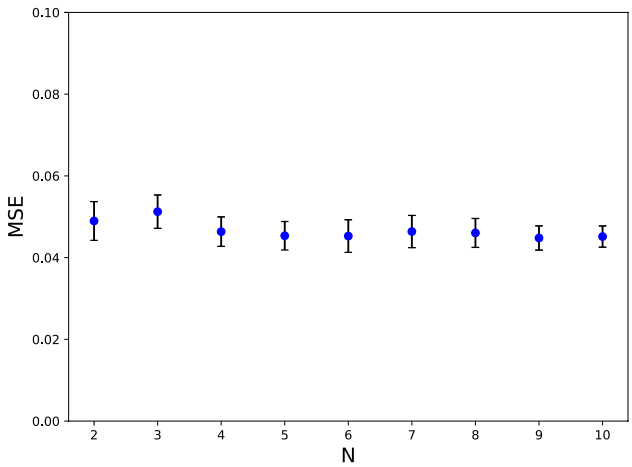

Figure 6. *MSE* between true and predicted potential as a function of charged particles number; in 100 different problems.

Methods

This article aims to build a machine-learning model (ML-Model) to predict the potential of point-charged particles surrounded by conductive walls. The potential of charged particles is calculated by solving the Poisson equation, which can be written as[4]:

$$\nabla^2\phi = -\rho/\epsilon_0 = -\sum_{i=1}^{N} q_i\delta(x - x_{q_i})/\epsilon_0, \tag{1}$$

where ϕ is the potential and ρ is a charge distribution. The first and straightforward ML-Model that jumps to mind is a model that includes x_q and x as an input and $\phi(x_q, x)$ as an output. Here x_q is the position of a point-charged particle, x is the position in which we want to calculate the potential on it, and $\phi(x_q, x)$ is the corresponding potential. So the number of input features depends on the number of charged particles; for instance, in 3 dimensions, if there are N charged particles, the input features have to be $3 + 3 \times N$. Therefore, this kind of model could only predict the potential of fix number of charged particles. One of the applications of this method is simulating charged particles transport in conductive porous media. The number of pores is more than unity in many simulation setups. Therefore, one common scenario is training a separate PINN for each system pore. In this case, even though the total number of particles is fixed during the simulation, the number of charged particles in each pore varies, which has to be considered.

We use the PLT algorithm to transform the Poisson equation into the Laplace equation with new boundary conditions to overcome this problem. This algorithm will be discussed in more detail in the following subsection. So we can train a model which includes x and modified boundary conditions as input features and $\phi(\phi_b, x)$ as an output. We define the boundary conditions only on N_b fixed points on the boundary $\{\phi_1, \phi_2, \ldots, \phi_{N_b}\}$. In this case, with the PLT algorithm, we can build a model with a fixed number of input features that can predict any charged particles' potential. For further explanation, Fig. 7 shows the flowchart of this study.

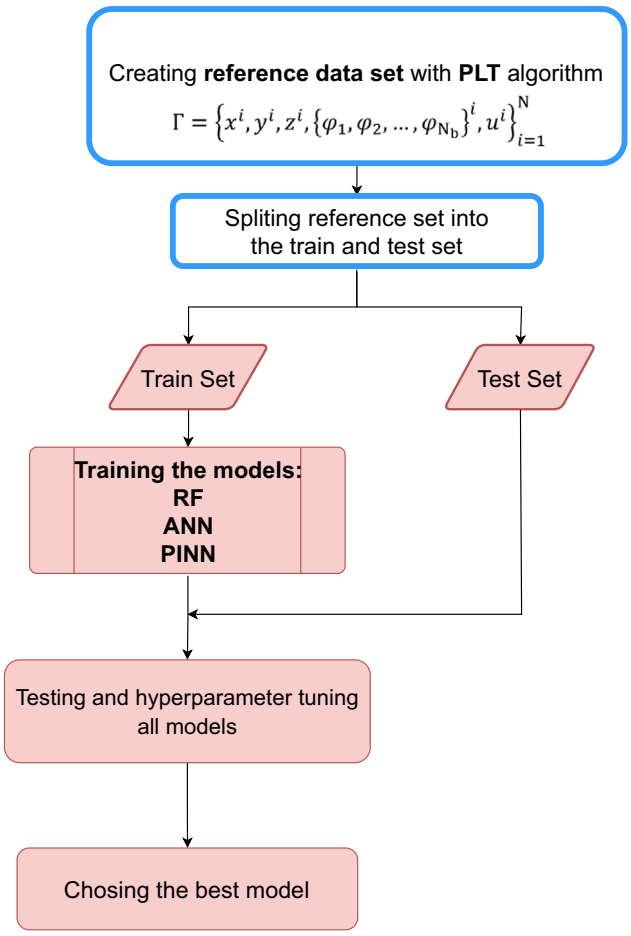

Figure 7. Methodology flow chart, The blue part: Preparing the data, in which the reference data set is created based on the PLT algorithm. The red part: Training models process, first the reference set is split to train and test set, then RF, ANN, and PINN model were applied on the train set, after tuning the hyperparameters the best model were chose.

Poisson to Laplace transformation (PLT). According to the PLT algorithm, the electrical potential is divided into two parts: singular potential (ϕ_{si}) and smooth potential (ϕ_{sm}); $\phi(\vec{x}) = \phi_{si}(\vec{x}) + \phi_{sm}(\vec{x})$. It is important to note that the smooth part here is the solution of the Laplace equation with modified boundary conditions,

$$\nabla^2 \phi_{sm}(\vec{x}) = 0, \tag{2}$$

while ϕ_{si} obeys the famous Coulomb's law

$$\phi_{si}(\vec{x}) = \sum_{i=1}^{N} \frac{q_i}{4\pi\epsilon_0 \|\vec{x} - \vec{x}_i\|}. \tag{3}$$

It can be seen that the modified boundary condition for ϕ_{sm} is represented by

$$\phi_{sm}\|_{\vec{x}_{bc}} = \phi\|_{\vec{x}_{bc}} - \phi_{si}\|_{\vec{x}_{bc}}, \tag{4}$$

where $\phi\|_{\vec{x}_{bc}}$ corresponds to the initial electrical potential on the boundaries. Figure 8 shows a schematic of the PLT algorithm. Finally, with the PLT algorithm, we could transfer the Poisson to the Laplace equation with new modified boundary conditions, then train an ML-Model with these modified boundary conditions as an input parameter and the smooth potential as an output. Afterward, we can reach the total potential with the summation of singular and predicted smooth potential. The advantage of utilizing the PLT algorithm is that it leads to having a fixed number of input data since the number of input data would be independent of the number of point-charged particles.

Data engineering. For training a highly accurate model, having a nice train set is crucial. In this work, the reference set is $\Gamma = \left\{x^i, y^i, z^i, \vec{\varphi}^i_{bc}, \phi^i\right\}_{i=1}^{N}$, where the input is concluded $\left\{x, y, z, \vec{\varphi}_{bc} = \left\{\phi_1, \phi_2, \ldots, \phi_{N_b}\right\}\right\}$ and ϕ is the target. x, y, z is the coordinate of a point in the container on which we want to calculate their potential, ϕ is the numeric value of the potential at this point, and $\left\{\phi_1, \phi_2, \ldots, \phi_{N_b}\right\}$ is the boundary condition on N_b points on the boundary. First, in the container, N_p positions are chosen to predict the potential in their situations. In fact, for each boundary condition, $\left\{\phi_1, \phi_2, \ldots, \phi_{N_b}\right\}$, there are N_p points that we want to calculate the potential at their positions. Then, the reference set could be created for N_q different boundary conditions. So the reference set consists of $N = N_q \times N_p$ samples which could split to train and test set. In this case, our container is a sphere; we also set $N_p = 78$, $N_b = 26$, and $N_q = 100$. So, our reference consists of 100 different boundary conditions and for each boundary condition $\{\phi_1, \phi_2, \ldots, \phi_{26}\}$ there are 78 points in the sphere on which we want to calculate the potential on it. We use the solution of the image charges method (Eq. 5) to calculate targets of the reference set:

$$\phi(\vec{x}) = \frac{1}{4\pi\epsilon_0}\left\{\frac{q}{\|\vec{x} - \vec{x}_q\|} + \frac{q'}{\|\vec{x} - \vec{x_{q'}}\|}\right\}, \quad q' = -\frac{rq}{a}, \quad \vec{x_{q'}} = \frac{a^2}{r}\frac{\vec{x}_q}{\|\vec{x}_q\|}, \tag{5}$$

where a is the conductive spherical shell radius and r is the distance of a point charge q from its center. The numeric value of potential is minimal ($\sim 10^{-9}$), which conducts to significant rounding error during computation; therefore, the potential of an electron in a 1 m distance of it, $1.44 \times 10^{-9}[V]$, is used as a unit to make Eq. (5) dimensionless. We randomly chose 5000 and 1000 samples from the reference set to create a train and test set. The train and test set have no samples in common. In addition, the best model could adequately predict the potential of test samples and samples with different boundary conditions from the train and test set. So to evaluate the model better, we prepare an extrapolation set that includes 1000 samples with 55 modified boundary conditions due to different configurations of charged particles.

ML algorithms. In this work, three different supervised learning methods have been used, and their regression accuracy, based on the metrics presented in the "Evaluating metrics" subsection, has been evaluated. Mainly, we stick to Physics-Informed Neural Networks (PINN[16]), but to compare our results with other ML algorithms,

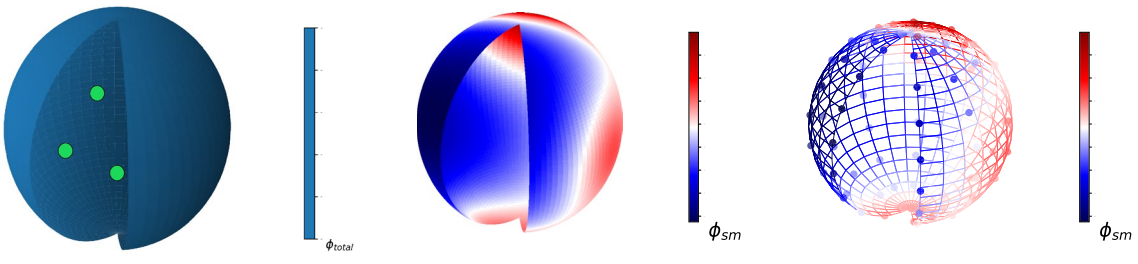

Figure 8. Schematic of the PLT method: (**a**) the main system which had point charges inside of it, (**b**) the new system without any point charges and the boundaries were modified, (**c**) N_b points on the boundary are shown to be used as our model input.

we use Random Forest (RF[17]) and Artificial Neural Networks (ANN). All the models are briefly introduced, the hyperparameters are fine-tuned, and their performance is reported. Scikit-learn[18], Tensorflow[19], Keras[20], and NumPy[21] are all the Python libraries that have been used in this project.

Random forest (RF). RF is one of the most popular machine learning algorithms in regression problems for many reasons, but this model has been chosen in this project since

(a) It is speedy to learn.
(b) It is robust against over-fitting.

Over-fitting is detected when the performance of train samples is perfect while the performance of test samples is poor. RF is an ensemble model in which an average of many uncorrelated trees determines the predicted potential for the target data set. Although each tree is a weak learner, they make a strong learner when many trees are grouped. The RF randomizes the trees by choosing a subset of training data and features for each tree. Here we use scikit-learn[18] RF implementation.

ANN. Typical neural network architecture consists of the input layer, multiple hidden layers, and the output layer with several neurons in each layer. Totally:

- Input layer: The neurons in the input layer are the input features.
- Hidden layers: The value of every neuron in the hidden layers is a linear combination of the neurons in the previous layer followed by the implementation of an activation function (Eq. 6); in most cases, the activation function is non-linear.

$$a_n = \sigma_l(a_{n-1}\mathbf{w}_n + \mathbf{b}_n). \tag{6}$$

  n is the layer number, **w** and **b** are the model parameters, weights and bias respectively, and σ_l is the activation function based on[22].
- Output layer: The neurons in the output layer are the model targets and they are calculated with Eq. (6) with linear activation function.
- Loss function: There is a function in all neural networks that must be minimized over the model parameters during the training stage via back-propagation, typically the loss function is the mean square error between the true and the predicted values.

$$\text{Loss}(w) = MSE_d + \lambda \sum_w w^2, \tag{7}$$

$$MSE_d = \frac{1}{N}\sum_{\mathrm{i}=1}^{\mathrm{N}} [\mathrm{U}(\mathbf{X}_\mathrm{i}, \mathbf{w}) - \mathrm{T}_\mathrm{i}]^2. \tag{8}$$

U and T are the predicted output and true target values, respectively, X is the input data, and w is the parameter of neural networks, weights, and biases. The first sentence in Eq. (7) is a mean square error, and The second sentence exists to prevent over-fitting, namely *L*2 regularization[23], that is used in order to reduce the effects of the large weights.

PINN. PINN[16] enforces the Laplace equation, a physical law of the electromagnetic system, as a constraint on the neural network. This study proposes a PINN-based approach to solve the Laplace equation with changeable boundary conditions. Figure 9 shows a schematic of the neural network layout for this approach. PINN-based models are neural networks with modified *loss* functions:

$$\text{Loss} = \lambda_1 MSE_d + \lambda_2 MSE_f + \lambda_3 MSE_b + \lambda_4 \sum_w w^2, \tag{9}$$

The first and the last term of Eq. (9) are the same as typical neural networks in Eq. (7). The second term corresponds to the governing physical equation, i.e., the is Laplace, and the third term corresponds to the boundary conditions;

$$MSE_d = \frac{1}{N_d}\sum_{i=1}^{N_d} \left\| u\left(\mathbf{x}_d^i, \vec{\varphi_d^i}; w\right) - \phi_d^i \right\|^2, \tag{10}$$

$$MSE_f = \frac{1}{N_f}\sum_{i=1}^{N_f} \left\| f\left(\mathbf{x}_f^i, u_f^i; w\right) \right\|^2, \tag{11}$$

and

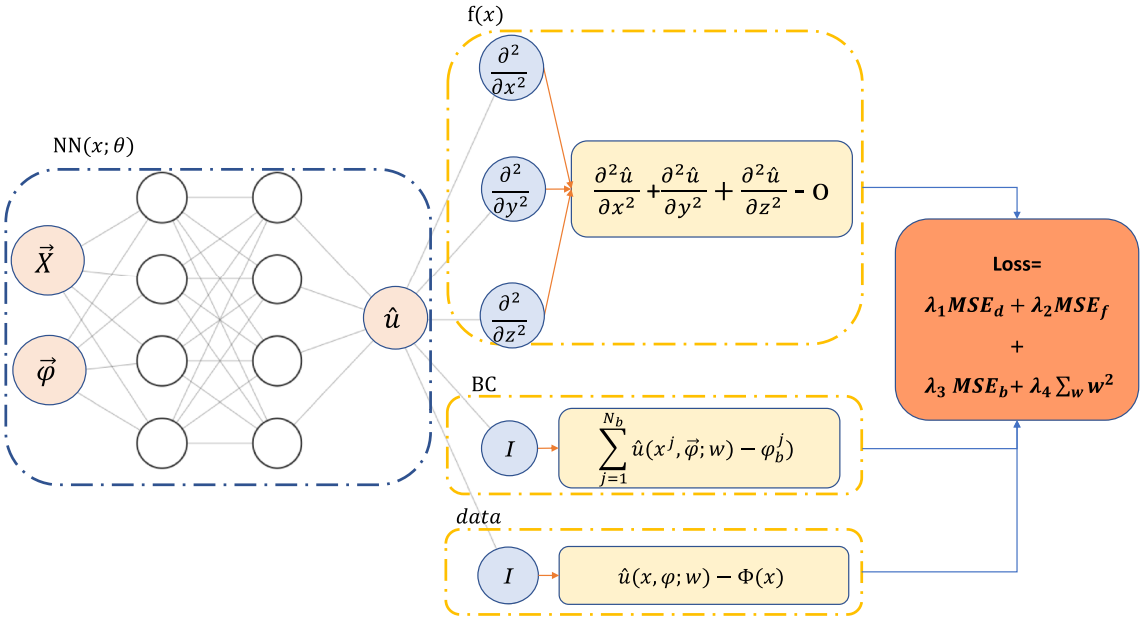

Figure 9. Physics-informed neural network scheme for solving Laplace equation with variable boundaries.

$$MSE_b = \frac{1}{N_b}\sum_{i=1}^{N_b}\left\|\mathcal{B}\left(\mathbf{x}_b^i, \vec{\varphi}_b^i, u_b^i; w\right)\right\|^2. \tag{12}$$

Here we define $f(\mathbf{x}, \mathrm{u}; \mathrm{w})$

$$\begin{aligned} f(\mathbf{x}, \mathrm{u}; \mathrm{w}) &= 0, \qquad \mathrm{x} \in \Gamma_\mathrm{f} \\ &= \nabla^2 u \\ &= \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2} \\ &= \frac{\partial w}{\partial x}\frac{\partial}{\partial w}\left(\frac{\partial w}{\partial x}\frac{\partial u}{\partial w}\right) + \frac{\partial w}{\partial y}\frac{\partial}{\partial w}\left(\frac{\partial w}{\partial y}\frac{\partial u}{\partial w}\right) + \frac{\partial w}{\partial z}\frac{\partial}{\partial w}\left(\frac{\partial w}{\partial z}\frac{\partial u}{\partial w}\right), \end{aligned} \tag{13}$$

with Dirichlet boundary conditions

$$\begin{aligned} \mathcal{B}(\mathbf{x}, \vec{\varphi}, u; w) &= 0, \qquad \mathrm{x} \in \Gamma_\mathrm{b} \\ &= \sum_{j=1}^{26}\left(u\left(\mathbf{x}^j, \vec{\varphi}; w\right) - \varphi_b^j\right). \end{aligned} \tag{14}$$

$\lambda_1, \lambda_2, \lambda_3$ in Eq. (9) correspond to the weight coefficients for the data contributions, Laplace equation, and boundary losses. We use the weight coefficient by motivating from the study of Kag et al.[24]. The last sentence is the *L2* regularization[23]. Notice that the model with $\lambda_2 = \lambda_3 = 0.0$ is exactly a typical neural network described in the previous subsection.

Evaluating metrics. The performance evaluation of different algorithms for potential estimation depends on different metrics,

$$\Delta\phi^i = \phi^i_{True} - \phi^i_{Pred}, \tag{15}$$

$$\sigma = \sqrt{\frac{1}{n}\sum_{i=0}^{n-1}(\Delta\phi^i)^2}, \tag{16}$$

$$\mathrm{R}^2 = 1 - \frac{\sum_{i=1}^{n}\left(\phi^i_{\mathrm{True}} - \phi^i_{\mathrm{Pred}}\right)^2}{\sum_{i=1}^{n}\left(\phi^i_{\mathrm{True}} - \bar{\phi}_{\mathrm{True}}\right)^2}, \tag{17}$$

$$\mathrm{MSE} = \langle(\Delta\phi)^2\rangle. \tag{18}$$

Where ϕ_{True} is the true potential, ϕ_{Pred} is the predicted potential, and $\bar{\phi}_{True}$ is the mean true potential of a given test sample. In this study we used scatter σ, R^2 score and *MSE* as our evaluating metrics.

Data availability

A complete description of how the datasets for the current study were generated can be found in the section of "Data Engineering". In addition, the datasets used and/or analyzed during the current study are available from the corresponding author upon reasonable request.

Received: 13 April 2023; Accepted: 10 August 2023
Published online: 28 August 2023

References

1. Miller, J. R. & Simon, P. Electrochemical capacitors for energy management. *Science* **321**(5889), 651–652 (2008).
2. Salanne, M. *et al.* Efficient storage mechanisms for building better supercapacitors. *Nat. Energy* **1**(6), 1–10 (2016).
3. Simon, P. & Gogotsi, Y. Materials for electrochemical capacitors. *Nat. Mater.* **7**(11), 845–854. https://doi.org/10.1038/nmat2297 (2008).
4. Jackson, J. D. *Classical Electrodynamics* (Wiley, 1975).
5. Jin, J. M. *The Finite Element Method in Electromagnetics* 3rd edn. (Wiley, 2015).
6. Golub, G. H. & Van Loan, C. F. *Matrix Computations* (JHU Press, 2013).
7. Tyagi, S., Arnold, A. & Holm, C. Icmmm2d: An accurate method to include planar dielectric interfaces via image charge summation. *J. Chem. Phys.* **127**, 154723. https://doi.org/10.1063/1.2790428 (2007).
8. Tyagi, S., Arnold, A. & Holm, C. Electrostatic layer correction with image charges: a linear scaling method to treat slab 2d+h systems with dielectric interfaces. *J. Chem. Phys.* **129**, 204102. https://doi.org/10.1063/1.3021064 (2008).
9. Tyagi, S. *et al.* An iterative, fast, linear-scaling method for computing induced charges on arbitrary dielectric boundaries. *J. Chem. Phys.* **132**, 154112. https://doi.org/10.1063/1.3376011 (2010).
10. Kesselheim, S., Sega, M. & Holm, C. The icc* algorithm: A fast way to include dielectric boundary effects into molecular dynamics simulations. arXiv:1003.1271 (2010).
11. Arnold, A. *et al.* Espresso 3.1: Molecular dynamics software for coarse-grained models, in *Meshfree Methods for Partial Differential Equations VI* 1–23. (Springer, 2013). https://doi.org/10.1007/978-3-642-32979-1_1
12. Reed, S. K., Lanning, O. J. & Madden, P. A. Electrochemical interface between an ionic liquid and a model metallic electrode. *J. Chem. Phys.* **126**, 084704. https://doi.org/10.1063/1.2464084 (2007).
13. Rostami, S., Ghasemi, S. A. & Nedaaee Oskoee, E. A highly accurate and efficient algorithm for electrostatic interactions of charged particles confined by parallel metallic plates. *J. Chem. Phys.* **145**(12), 124118 (2016).
14. Biagooi, M., Samanipour, M., Ghasemi, S. A. & Oskoee, S. N. Caviar: A simulation package for charged particles in environments surrounded by conductive boundaries. *AIP Adv.* **10,** 035310. https://doi.org/10.1063/1.5140052 (2020).
15. Shan, T. *et al.* Study on a fast solver for Poisson's equation based on deep learning technique. *IEEE Trans. Antennas Propag.* **68**(9), 6725–6733. https://doi.org/10.1109/TAP.2020.2985172 (2020).
16. Raissi, M., Perdikaris, P. & Karniadakis, G. E. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *J. Comput. Phys.* **378**, 686–707. https://doi.org/10.1016/j.jcp.2018.10.045 (2019).
17. Breiman, L. Random forests. *Mach. Learn.* **45**. https://doi.org/10.1023/A:1010933404324 (2001).
18. Pedregosa, F. *et al.* Scikit-learn: Machine learning in python. *J. Mach. Learn. Res.* **12**, 2825–2830 (2011).
19. Abadi, M. Tensorflow: Learning functions at scale. *ACM SIGPLAN Not.* **51**. https://doi.org/10.1145/3022670.2976746 (2016).
20. Chollet, F. *Keras: The Python Deep Learning Library* (Keras.Io, 2015).
21. Walt, S. V. D., Colbert, S. C. & Varoquaux, G. The numpy array: A structure for efficient numerical computation. *Comput. Sci. Eng.* **13**, 22–30. https://doi.org/10.1109/MCSE.2011.37 (2011).
22. Goodfellow, I., Bengio, Y. & Courville, A. *Deep Learning* (MIT Press, 2016). http://www.deeplearningbook.org
23. Krogh, A. & Hertz, J. A. A simple weight decay can improve generalization. *Adv. Neural Inf. Process. Syst.* **4**, 950–957 (1992).
24. Kag, V., Seshasayanan, K. & Gopinath, V. Physics and data informed neural networks for two-dimensional turbulence. *Phys. Fluids* **34**(5), 055130 (2022).
25. Liu, D. C. & Nocedal, J. On the limited memory BFGS method for large scale optimization. *Math. Program.* **45**(1), 503–528 (1989).
26. Eyvazi, N., Biagooi, M. & Nedaaee Oskoee, E. Molecular dynamics investigation of charging process in polyelectrolyte-based supercapacitors. *Sci. Rep.* **12**(1), 1098 (2022).

Author contributions

F.H. designed the machine learning model, analyzed the results, and drafted the manuscript. M.B. collected and analyzed the initial data points. S.E.N.O. reanalyzed the results and revised the manuscript. F.H. and S.E.N.O. defined the problem and all of the authors reviewed the final manuscript.

Competing interests

The authors declare no competing interests.

Additional information

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1038/s41598-023-40477-y.

Correspondence and requests for materials should be addressed to S.N.O.