Neural Network Poisson–Boltzmann Electrostatics for Biomolecular Interactions

Zunding Huang[1], Bo Li[2,5], Zhongming Wang[3], and Zhiwen Zhang[4,5]

[1] Department of Mathematics, University of California, San Diego, La Jolla, California, USA. Email: zuhuang@ucsd.edu

[2] Department of Mathematics and Quantitative Biology Ph.D. Program, University of California, San Diego, La Jolla, California, USA. Email: bli@math.ucsd.edu

[3] Department of Mathematics and Statistics, Florida International University, Miami, USA. Email: zwang6@fiu.edu

[4] Department of Mathematics, The University of Hong Kong, Hong Kong, China. Email: zhangzw@hku.hk

[5]Corresponding authors

October 8, 2025

Abstract

We develop a neural network approach to solving the dielectric-boundary Poisson–Boltzmann (PB) equation (PBE) and estimating the electrostatic free energy of charged molecules in aqueous solvent. Such equation is the Euler–Lagrange equation of the classical PB electrostatic free-energy functional with the presence of a dielectric boundary. We construct a penalized dielectric-boundary PB functional to remove the constraint imposed by the boundary condition for the boundary of the computational region and show that such penalized functionals converge to the classical PB functional. We represent electrostatic potentials by fully connected feed-forward neural network functions with sigmoidal activation, use the penalized functional and Monte Carlo integration method to construct a neural network loss function, and minimize it by a stochastic gradient-descent (SGD) method. Numerical results are presented to show the convergence of the neural network simulations with varying learning rates, batch size, and network architecture. Moreover, the relation between the boundedness of network weights and learning rates in the loss optimization is explored. The neural network PB method is applied to the calculation of electrostatic free energy of the solvation of single ions and protein BphC, demonstrating that the new approach can handle both simple and complex geometries and predict qualitatively well the electrostatic energy. In particular, we find that using the trained neural network weights from one simulation as the initial weights for simulations with different settings significantly increases the simulation efficiency. Such transferability of network weights provides an advantage of our neural network PB approach to complex biomolecular systems.

Keywords. Poisson–Boltzmann equation, biomolecular electrostatic energy, neural networks, Monte Carlo simulations, stochastic gradient descent, weight transferability.

1 Introduction

We develop a neural network approach to solving the dielectric boundary Poisson–Boltzmann (PB) equation (PBE) and estimating the electrostatic energy of the solvation of charged solute molecules such as single ions and proteins.

Let us denote by $\Omega \subset \mathbb{R}^3$ the region of solvation and by Γ a smooth and closed surface that divides Ω into the solute molecular region Ω_- and the solvent region Ω_+, inside and outside Γ, respectively; cf. Figure 1. The solute region Ω_-, which may have multiple connected components, contains solute atoms located at $x_1, \dots, x_N$ and carrying charges $Q_1, \dots, Q_N$, respectively. Mobile ions in the solvent are assumed to be excluded from the solute region. We denote by ε_- and ε_+, two positive numbers, the dielectric coefficients of the solute region Ω_- and the solvent region Ω_+, respectively. Typically, $\varepsilon_- \approx 1$ and $\varepsilon_+ \approx 80$ in the unit of vacuum permittivity. We call Γ a solute-solvent interface or a dielectric boundary.

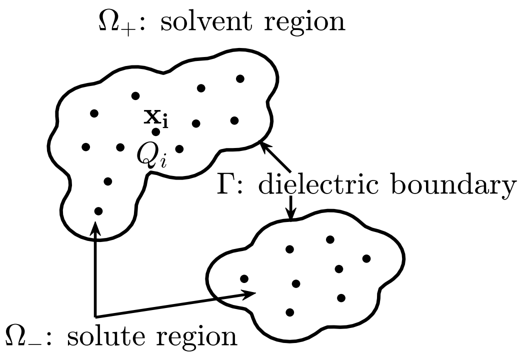
Ω_+: solvent region
$\mathbf{x_i}$
Q_i
Γ: dielectric boundary
Ω_-: solute region

Figure 1: A schematic diagram of charged molecules immersed in aqueous solvent, where Ω is divided into the solute and solvent regions Ω_- and Ω_+, respectively, by the solute-solvent interface or dielectric boundary Γ. Dots represent solute atoms.

In a continuum model, the equilibrium electrostatic potential $\phi_\Gamma : \Omega \to \mathbb{R}$ is uniquely determined by the dielectric-boundary PBE [6, 10, 37]

$$\nabla \cdot \varepsilon_\Gamma \nabla \phi_\Gamma - \chi_+ B'(\phi_\Gamma) = -f \qquad \text{in } \Omega, \tag{1.1}$$

together with some boundary conditions, e.g., $\phi_\Gamma = g$ on $\partial\Omega$ for some given function g, when Ω is chosen to be a bounded region. Here and below, $\varepsilon_\Gamma : \Omega \to \mathbb{R}$ is the dielectric coefficient, defined by

$$\varepsilon_\Gamma = \begin{cases} \varepsilon_- & \text{in } \Omega_-, \\ \varepsilon_+ & \text{in } \Omega_+, \end{cases} \tag{1.2}$$

$\chi_+ = \chi_{\Omega_+}$ denotes the indicator function of Ω_+: $\chi_+(x) = 1$ if $x \in \Omega_+$ and $\chi_+(x) = 0$ otherwise, and the function f approximates the point charges $\sum_{i=1}^N Q_i \delta_{x_i}$, where δ_a denotes the Dirac measure centered at $a \in \mathbb{R}^3$. The term $-B'(\phi_\Gamma)$ describes the charge density of mobile ions in the solvent. Typically, the function $B : \mathbb{R} \to \mathbb{R}$ is given by

$$B(\phi) = \beta^{-1} \sum_{j=1}^{M} c_j^\infty (e^{-\beta q_j \phi} - 1), \tag{1.3}$$

where $M \geq 1$ is the number of ionic species, $\beta^{-1} = k_B T$ with k_B the Boltzmann constant and T temperature, $q_j = Z_j e$ with e the elementary charge, and c_j^∞ and Z_j are the bulk ionic concentration and valence of ions of the jth ionic species, respectively. Note that the function B is smooth, convex, and bounded below. We shall assume some $q_j > 0$ and some other $q_j < 0$. Consequently, we have $B(\infty) = B(-\infty) = \infty, B'(-\infty) = -\infty$, and $B'(\infty) = \infty$. Two commonly used forms of the function B are: (1) A hyperbolic cosine function

$$B(\phi) = 2\beta^{-1} c^\infty \left[\cosh(\beta\phi) - 1\right],$$

which models a $1:1$ salt with the same charge $q_1 = -q_2 = 1$ and the same bulk ionic concentration $c_1^\infty = c_2^\infty = c^\infty$; and (2) The linearized PB model for a charge-neutralized ionic solution (i.e., $\sum_{k=1}^M q_k c_k^\infty = 0$)

$$B(\phi) = \frac{\beta}{2}\left(\sum_{j=1}^M c_j^\infty q_j^2\right)\phi^2.$$

There are two distinguished features of the PBE (1.1). One is the nonlinearity and the other is the specific piecewise constant structure of ε_Γ due to the presence of the dielectric boundary Γ. However, it is known that the PBE (1.1) is the Euler–Lagrange equation of the classical PB electrostatic energy functional. This variational structure helps the mathematical analysis of the well-posedness and other properties of the PBE [22, 23, 24]. Many numerical methods have been developed to solve the boundary-value problem of the PBE. Finite-element methods are based on the weak formulation of the PBE. Moreover, within such a framework, the nonlinearity is often treated using Newton's iteration [29, 32]. Most finite-difference methods are designed for solving the equivalent, elliptic interface problem [9, 34, 44]

$$\begin{cases} \varepsilon_- \Delta\phi = -f & \text{in } \Omega_-, \\ \varepsilon_+ \Delta\phi - B'(\phi) = -f & \text{in } \Omega_+, \\ [\![\phi]\!] = [\![\varepsilon_\Gamma \nabla\phi \cdot n]\!] = 0 & \text{on } \Gamma, \\ \phi = g & \text{on } \partial\Omega, \end{cases} \tag{1.4}$$

where $[\![u]\!] = u|_{\Omega_+} - u|_{\Omega_-}$ denotes the jump of a function u across the dielectric boundary Γ and n denotes the unit normal at the dielectric boundary Γ pointing from Ω_- to Ω_+. The linearized PBE can be reformulated using boundary integrals, and the related fast algorithms have been well developed [28, 45].

In this work, we explore the ability of neural networks in simulating complex biological molecular systems with solving the dielectric-boundary PBE being the first step. We utilize the variational structure of our underlying system and follow the Deep-Ritz framework [13] (cf. also [26]) with certain modifications. The key components of our neural network approach include:

(1) We reformulate the problem of solving the dielectric-boundary PBE (1.1) into a variational problem of minimizing a penalized dielectric-boundary PB electrostatic free-energy functional. We shall show the convergence of such penalized functionals to the classical PB functional;
(2) We represent admissible electrostatic potentials by fully connected feed-forward neural network functions with a fixed network architecture and construct from the penalized functional a neural network loss function by Monte Carlo integration; and
(3) We minimize the loss function using a method of stochastic gradient descent (SGD).

We test the convergence and efficiency of our neural network method in terms of learning rates, batch size, and network architecture. We also examine the growth of network weights during the SGD iteration. We apply our approach to estimating of solvation electrostatic free energies for single ions and also for the protein BphC. Such application is closely connected to an advanced, variational implicit-solvent model (VISM) [11, 12, 42, 47] that determines the dielectric boundaries and provides estimates of the solvation free energies for charged molecules in aqueous solvent. We examine the transferability of network weights to increase the simulation efficiency demanded by large-scale and multiple simulations in these applications.

There has been a growing interest in using neural networks for solving partial differential equations (PDEs), but it is speculated that such an approach may only be effective for high-dimensional PDEs [15, 16, 38]. In general, classical numerical methods, such as finite-element, finite-difference, and boundary-integral methods, can solve low-dimensional PDEs very accurately and efficiently. We are motivated, however, by some of the progress in using neural networks to solve low-dimensional PDEs. For instance, such a method can treat complex domains and also interface problems with very high contrast, which corresponds to $\varepsilon_+/\varepsilon_-$ being very large for our case [3, 18, 40, 43]. Moreover, dielectric boundaries of charged molecules can be very complicated. As modeling and simulations of biomolecular systems remain a highly challenging scientific problem, it is natural to explore a new, machine-learning approach to complex biomolecular systems. We notice that a multiscale neural network approach is developed in [27] for solving the PB equation. In [21], the authors developed a neural network method for solving the PB equation using the interface formulation (1.4).

The rest of the paper is organized as follows: In Section 2, we present the dielectric-boundary PBE and the penalized PB electrostatic free-energy functional. We also prove the convergence of the penalized functional. In Section 3, we introduce our neural network structures, construct a loss function based on the Monte Carlo approximation of the penalized PB functional, and detail a SGD method for minimizing the loss function. In Section 4, we present numerical results to demonstrate the performance of our neural network simulations with respect to the learning rate, batch size, and network architecture. We also validate numerically our penalty method and examine the growth of network weights during the SGD iteration. In Section 5, we apply our neural network approach to the solvation of charged molecules, such as single ions and the protein BphC, in aqueous solvent. We particularly present results of simulated potential of mean electrostatic force between two domains of BphC. In addition, we show the strong transferability of network weights in these simulations. Finally, in Section 6, we conclude our studies with some discussions.

2 The Dielectric-Boundary Poisson–Boltzmann Equation and a Penalized Functional

Let us assume the system region Ω is bounded and regular (e.g., its boundary $\partial\Omega$ is Lipschitz-continuous). We fix $g \in H^1(\Omega)$ and denote $H^1_g(\Omega) = \{\phi \in H^1(\Omega) : \phi = g \text{ on } \partial\Omega\}$. (The space $H^1(\Omega)$ consists of all functions u on Ω such that u^2 and $|\nabla u|^2$ are both integrable on Ω [1, 14].) We introduce the functional $I_\Gamma : H^1_g(\Omega) \to \mathbb{R} \cup \{+\infty\}$ by

$$I_\Gamma[\phi] = \int_\Omega \left[\frac{\varepsilon_\Gamma}{2} |\nabla\phi|^2 - f\phi + \chi_+ B(\phi) \right] dx. \tag{2.1}$$

Note that this is a convex functional and its Euler–Lagrange equation is exactly the dielectric-boundary PBE (1.1). If $\phi_\Gamma \in H^1_g(\Omega)$ is the (weak) solution to the dielectric-boundary PBE (1.1), then ϕ_Γ is the unique minimizer of $I_\Gamma : H^1_g(\Omega) \to \mathbb{R} \cup \{+\infty\}$; cf. Theorem 2.1 below which is proved in [23, 24, 25]. Moreover, the electrostatic energy is given by [6]

$$E_{\rm ele}[\Gamma] = -I_\Gamma[\phi_\Gamma] = \max_{\phi \in H^1_g(\Omega)} -I_\Gamma[\phi]. \tag{2.2}$$

We refer to [6] for discussions on the negative sign.

Theorem 2.1. *Let $\Omega \subset \mathbb{R}^3$ be a bounded domain with a C^1 boundary $\partial\Omega$. and Γ as above be a C^1 closed surface. Let $f \in L^2(\Omega)$ and $g \in W^{1,\infty}(\Omega)$. Then there exists a unique $\phi_\Gamma \in H^1_g(\Omega)$ such that $I_\Gamma[\phi_\Gamma] = \min_{\phi \in H^1_g(\Omega)} I_\Gamma[\phi]$ which is finite. Moreover, $\phi_\Gamma \in L^\infty(\Omega)$ and ϕ_Γ is also the unique weak solution to the PBE* (1.1)*. Equivalently, ϕ_Γ is the unique solution to the system of equations* (1.4)*.* $\square$

Let $\lambda > 0$. We define the penalized PB electrostatic free-energy functional $I_{\Gamma,\lambda} : H^1(\Omega) \to \mathbb{R} \cup \{+\infty\}$ by

$$I_{\Gamma,\lambda}[\phi] = \int_\Omega \left[\frac{\varepsilon_\Gamma}{2} |\nabla \phi|^2 - f\phi + \chi_+ B(\phi) \right] dx + \lambda \int_{\partial\Omega} (\phi - g)^2 \, dS \qquad \forall \phi \in H^1(\Omega). \tag{2.3}$$

Note that we approximate the electrostatic energy by $\sup_{\phi \in H^1(\Omega)}(-I_{\Gamma,\lambda}[\phi])$ with a large $\lambda > 0$; cf. (2.2). The following theorem indicates that the penalty functional approximates the PB functional well for a large penalty parameter $\lambda > 0$.

Theorem 2.2. *Let Ω, Γ, f, and g be the same as in Theorem 2.1. For any $\lambda > 0$, there exists a unique $\phi_{\Gamma,\lambda} \in H^1(\Omega)$ such that $I_{\Gamma,\lambda}[\phi_{\Gamma,\lambda}] = \min_{\phi \in H^1(\Omega)} I_{\Gamma,\lambda}[\phi]$. Moreover, $\phi_{\Gamma,\lambda} \to \phi_\Gamma$ in $H^1(\Omega)$ and $\min_{\phi \in H^1(\Omega)} I_{\Gamma,\lambda}[\phi] \to \min_{\phi \in H^1_g(\Omega)} I_\Gamma[\phi]$ as $\lambda \to +\infty$.*

Proof. Fix $\lambda > 0$. Since $B : \mathbb{R} \to \mathbb{R}$ defined in (1.3) is bounded below, we have by the definition of ε_Γ (cf. (1.2)) and $I_{\Gamma,\lambda} : H^1(\Omega) \to \mathbb{R} \cup \{+\infty\}$ (cf. (2.3)), and Friedrichs' inequality [39] that

$$I_{\Gamma,\lambda}[\phi] \geq C_1 \|\phi\|^2_{H^1(\Omega)} - C_2 \qquad \forall \phi \in H^1(\Omega), \tag{2.4}$$

where $C_1 > 0$ and $C_2 > 0$ are two constants that may depend on ε_-, ε_+, f, g, λ, B, Ω, and Γ.

Now, let $\alpha_{\Gamma,\lambda} = \inf_{\phi \in H^1(\Omega)} I_{\Gamma,\lambda}[\phi]$. By (2.4), it is clear that $\alpha_{\Gamma,\lambda} \in \mathbb{R}$. Moreover, there exist $\phi_k \in H^1(\Omega)$ $(k = 1, 2, \dots)$ such that $I_{\Gamma,\lambda}[\phi_k] \to \alpha_{\Gamma,\lambda}$ as $k \to \infty$. It follows from (2.4) that, there exist a subsequence of $\{\phi_k\}$, not relabeled, and some $\phi_{\Gamma,\lambda} \in H^1(\Omega)$ such that $\phi_k \to \phi_{\Gamma,\lambda}$ weakly in $H^1(\Omega)$, strongly in $L^2(\Omega)$, and strongly in $L^2(\partial\Omega)$, and almost everywhere in Ω. These convergence results and Fatou's lemma imply that,

$$\begin{aligned}
&\liminf_{k\to\infty} \left[\int_\Omega \left(\frac{\varepsilon_\Gamma}{2} |\nabla \phi_k|^2 - f\phi_k \right) dx + \lambda \int_{\partial\Omega} (\phi_k - g)^2 \, dS \right] \\
&\qquad \geq \int_\Omega \left(\frac{\varepsilon_\Gamma}{2} |\nabla \phi_{\Gamma,\lambda}|^2 - f\phi_{\Gamma,\lambda} \right) dx + \lambda \int_{\partial\Omega} (\phi_{\Gamma,\lambda} - g)^2 \, dS, \\
&\liminf_{k\to\infty} \int_\Omega \chi_+ B(\phi_k) \, dx \geq \int_\Omega \chi_+ B(\phi_{\Gamma,\lambda}) \, dx.
\end{aligned}$$

Consequently, $\alpha_{\Gamma,\lambda} \geq I_{\Gamma,\lambda}[\phi_{\Gamma,\lambda}] \geq \alpha_{\Gamma,\lambda}$, and $I_{\Gamma,\lambda}[\phi_{\Gamma,\lambda}] = \min_{\phi\in H^1(\Omega)} I_{\Gamma,\lambda}[\phi]$. Since $I_{\Gamma,\lambda} : H^1(\Omega) \to \mathbb{R}\cup\{+\infty\}$ is strictly convex, the minimizer $\phi_{\Gamma,\lambda} \in H^1(\Omega)$ of $I_{\Gamma,\lambda} : H^1(\Omega) \to \mathbb{R}$ is unique.

It follows from the definition of $I_{\Gamma,\lambda} : H^1(\Omega) \to \mathbb{R}\cup\{+\infty\}$ that $I_{\Gamma,\lambda}[\phi]$ increases as $\lambda > 0$ increases for each $\phi \in H^1(\Omega)$, and hence $\alpha_{\Gamma,\lambda} = I_{\Gamma,\lambda}[\phi_{\Gamma,\lambda}] = \min_{\phi\in H^1(\Omega)} I_{\Gamma,\lambda}[\phi]$ is also an increasing function of λ. Moreover, since $g \in W^{1,\infty}(\Omega)$ and $I_{\Gamma,\lambda}[g] = I_\Gamma[g]$ is independent of λ, we have $\sup_{\lambda>0}\alpha_{\Gamma,\lambda} \leq I_\Gamma[g] < \infty$. Since B is bounded below, we have for some constant C independent of $\lambda > 1$ such that

$$\begin{aligned}\int_\Omega \left(\frac{\varepsilon_\Gamma}{2}|\nabla\phi_{\Gamma,\lambda}|^2 - f\phi_{\Gamma,\lambda}\right)dx + \int_{\partial\Omega}(\phi_{\Gamma,\lambda}-g)^2\,dS&\\ \leq I_{\Gamma,\lambda}[\phi_{\Gamma,\lambda}] + C = \alpha_{\Gamma,\lambda} + C \leq I_\Gamma[g] + C < \infty \qquad \forall\lambda > 1.&\end{aligned}$$

With this and Friedrichs' inequality, we infer that

$$\sup_{\lambda\geq 1}\|\phi_{\Gamma,\lambda}\|_{H^1(\Omega)} < \infty, \tag{2.5}$$

and further that

$$\sup_{\lambda\geq 1}\lambda\int_{\partial\Omega}(\phi_{\Gamma,\lambda}-g)^2\,dS < \infty. \tag{2.6}$$

Consequently,

$$\phi_{\Gamma,\lambda} \to g \qquad \text{in } L^2(\partial\Omega) \quad \text{as } \lambda\to\infty. \tag{2.7}$$

Given any $\{\phi_{\Gamma,\lambda_k}\}_{k=1}^\infty$ in $H^1(\Omega)$ with $0 < \lambda_k \to \infty$. By (2.5), there exists $u_\Gamma \in H^1(\Omega)$ and a possibly further subsequence, not relabelled, such that $\phi_{\Gamma,\lambda_k} \to u_\Gamma$ weakly in $H^1(\Omega)$, strongly in both $L^2(\Omega)$ and $L^2(\partial\Omega)$, and almost everywhere in Ω. Moreover, by the same argument used above, we have $\liminf_{k\to\infty} I_\Gamma[\phi_{\Gamma,\lambda_k}] \geq I_\Gamma[u_\Gamma]$. By (2.7), $u_\Gamma = g$ on $\partial\Omega$. Hence $u_\Gamma \in H^1_g(\Omega)$. Since $\phi_\Gamma \in H^1_g(\Omega)$ (cf. Theorem 2.1), $I_\Gamma[\phi_\Gamma] = I_{\Gamma,\lambda}[\phi_\Gamma] \geq I_{\Gamma,\lambda}[\phi_{\Gamma,\lambda}]$ for all $\lambda > 0$. Consequently,

$$I_\Gamma[\phi_\Gamma] \geq \limsup_{k\to\infty} I_{\Gamma,\lambda_k}[\phi_{\Gamma,\lambda_k}] \geq \liminf_{k\to\infty} I_{\Gamma,\lambda_k}[\phi_{\Gamma,\lambda_k}] \geq \liminf_{k\to\infty} I_\Gamma[\phi_{\Gamma,\lambda_k}] \geq I_\Gamma[u_\Gamma]. \tag{2.8}$$

Hence, u_Γ is also a minimizer of $I_\Gamma : H^1_g(\Omega) \to \mathbb{R}\cup\{+\infty\}$. Since such a minimizer is unique by Theorem 2.1, we have $u_\Gamma = \phi_\Gamma$. This and (2.8) imply that $I_{\Gamma,\lambda_k}[\phi_{\Gamma,\lambda_k}] \to I_\Gamma[\phi_\Gamma]$.

To finally show that $\phi_{\Gamma,\lambda_k} \to \phi_\Gamma = u_\Gamma$ strongly in $H^1(\Omega)$ as $k\to\infty$, we observe that

$$\begin{aligned}0 &\leq I_{\Gamma,\lambda_k}[\phi_\Gamma] - I_{\Gamma,\lambda_k}[\phi_{\Gamma,\lambda_k}] = I_\Gamma[\phi_\Gamma] - I_{\Gamma,\lambda_k}[\phi_{\Gamma,\lambda_k}] \leq I_\Gamma[\phi_\Gamma] - I_\Gamma[\phi_{\Gamma,\lambda_k}]\\ &= \int_\Omega \frac{\varepsilon_\Gamma}{2}\left(|\nabla\phi_\Gamma|^2 - |\nabla\phi_{\Gamma,\lambda_k}|^2\right)dx - \int_\Omega f\left(\phi_\Gamma - \phi_{\Gamma,\lambda_k}\right)dx + \int_{\Omega_+}\left[B(\phi_\Gamma) - B(\phi_{\Gamma,\lambda_k})\right]dx\\ &:= A_k + B_k + C_k \qquad \forall k\geq 1.\end{aligned}$$

Since $\phi_{\Gamma,\lambda_k} \to \phi_\Gamma$ weakly in $H^1(\Omega)$, we have

$$\begin{aligned}\limsup_{k\to\infty} A_k &= \limsup_{k\to\infty}\int_\Omega \frac{\varepsilon_\Gamma}{2}\left(|\nabla\phi_\Gamma|^2 - |\nabla\phi_{\Gamma,\lambda_k}|^2\right)dx\\ &= \limsup_{k\to\infty}\left[-\int_\Omega\frac{\varepsilon_\Gamma}{2}|\nabla\phi_{\Gamma,\lambda_k} - \nabla\phi_\Gamma|^2\,dx - \int_\Omega \varepsilon_\Gamma(\nabla\phi_{\Gamma,\lambda_k} - \nabla\phi_\Gamma)\cdot\nabla\phi_\Gamma\right]\\ &= -\liminf_{k\to\infty}\int_\Omega\frac{\varepsilon_\Gamma}{2}|\nabla\phi_{\Gamma,\lambda_k} - \nabla\phi_\Gamma|^2\,dx\end{aligned}$$

$$\leq 0,$$

and

$$\lim_{k\to\infty} B_k = \lim_{k\to\infty} \left[-\int_\Omega f(\phi_\Gamma - \phi_{\Gamma,\lambda_k})\, dx \right] = 0.$$

Since $B : \mathbb{R} \to \mathbb{R}$ (1.3) is smooth, strictly convex, and bounded below, $\phi_\Gamma \in L^\infty(\Omega)$ (cf. Theorem 2.1) which implies that $B'(\phi_\Gamma) \in L^\infty(\Omega_+)$, and $\phi_{\Gamma,\lambda_k} \to \phi_\Gamma$ weakly in $H^1(\Omega)$ and almost everywhere in Ω, we have

$$\limsup_{k\to\infty} C_k = \limsup_{k\to\infty} \int_{\Omega_+} \left[B(\phi_\Gamma) - B(\phi_{\Gamma,\lambda_k})\right] dx \leq \limsup_{k\to\infty} \int_{\Omega_+} B'(\phi_\Gamma)(\phi_\Gamma - \phi_{\Gamma,\lambda_k})\, dx = 0.$$

Consequently,

$$0 \leq \limsup_{k\to\infty}(A_k + B_k + C_k) \leq \limsup_{k\to\infty} A_k + \limsup_{k\to\infty} C_k \leq \limsup_{k\to\infty} A_k \leq 0,$$

and hence $\limsup_{k\to\infty} A_k = 0$. Passing to a further subsequence if necessary, we have $A_k \to 0$ as $k \to \infty$. This and the weak convergence $\phi_{\Gamma,\lambda_k} \rightharpoonup \phi_\Gamma$ in $H^1(\Omega)$ imply that $\nabla\phi_{\Gamma,\lambda_k} \to \nabla\phi_\Gamma$ in $[L^2(\Omega)]^3$. Since $\phi_{\Gamma,\lambda_k} \to \phi_\Gamma$ strongly in $L^2(\Omega)$, we have $\phi_{\Gamma,\lambda_k} \to \phi_\Gamma$ in $H^1(\Omega)$. Finally, the arbitrariness of the sequence $\lambda_k \to \infty$ implies the designed convergence. $\square$

3 Neural Networks, Loss Function, and Stochastic Optimization

Neural networks. We use the standard fully-connected feed-forward neural network (FFNN) functions to represent electrostatic potentials [5, 17]. Given positive integers $m, n_0, n_1, \ldots, n_{m+1}$, we denote $S = [n_0, \ldots, n_{m+1}]$. Given additionally a function $\sigma : \mathbb{R} \to \mathbb{R}$, we define $\Psi_{S,\sigma}$ to be the class of FFNN functions of $m + 1$ layers and the activation function σ, with an input layer, an output layer, and m hidden layers and with the dimension (i.e., the number of nodes or neurons) of these layers being n_0, n_{m+1}, and n_i $(1 \leq i \leq m)$, respectively. We call $S = [n_0, \ldots, n_{m+1}]$ the neural network architecture. Let us define

$$\Theta_S = \{\theta = (W_i, b_i)_{i=1}^{m+1} : W_i \in \mathbb{R}^{n_i \times n_{i-1}},\ b_i \in \mathbb{R}^{n_i},\ i = 1, \ldots, m+1\}, \tag{3.1}$$

where $\mathbb{R}^{p\times q}$ denotes the set of real $p \times q$ matrices. For each $\theta = (W_i, b_i)_{i=1}^{m+1} \in \Theta_S$, we define $\psi_\theta : \mathbb{R}^{n_0} \to \mathbb{R}^{n_{m+1}}$ by

$$\psi_\theta(\mathbf{x}) = T_{m+1} \circ \sigma \circ \cdots \circ \sigma \circ T_1(\mathbf{x}) = T_{m+1}(\sigma(\cdots(\sigma(T_1(\mathbf{x}))))) \qquad \forall \mathbf{x} \in \mathbb{R}^{n_0}, \tag{3.2}$$

where each $T_i : \mathbb{R}^{n_{i-1}} \to \mathbb{R}^{n_i}$ is an affine mapping defined by $T_i(\mathbf{x}) = W_i\mathbf{x} + b_i$ $(1 \leq i \leq m + 1)$. The class of FFNN functions $\Psi_{S,\sigma}$ is exactly defined by

$$\Psi_{S,\sigma} = \{\psi_\theta : \mathbb{R}^{n_0} \to \mathbb{R}^{n_{m+1}} : \theta \in \Theta_S\}.$$

If $\theta = (W_i, b_i)_{i=1}^{m+1} \in \Theta_S$, then all the entries of the matrices W_i and vectors b_i (or $-b_i$) ($i = 1, \ldots, m + 1$) are called, respectively, the weights and biases, or collectively just the weights or trainable parameters, of the neural network function ψ_θ.

Throughout, we set the input dimension $n_0 = 3$ and the output dimension $n_{m+1} = 1$, and we use the standard sigmoid function $\sigma(x) = 1/(1 + e^{-x})$ $(x \in \mathbb{R})$ as the activation function, unless otherwise stated. If $\mathbf{a} = (a_1, \ldots, a_k) \in \mathbb{R}^k$ for some integer $k \geq 1$, then we denote $\sigma(\mathbf{a}) = \big(\sigma(a_1), \ldots, \sigma(a_k)\big)$.

The neural network loss function and its gradient. We recall the penalized PB electrostatic energy functional $I_{\Gamma,\lambda} : H^1(\Omega) \to \mathbb{R} \cup \{+\infty\}$ defined in (2.3) with a fixed $\lambda > 0$ and all f, g, Ω, Ω_-, Ω_+, Γ, ε_Γ, and B given in Theorem 2.1 and Theorem 2.2. We note that, for a given network architecture $S = [n_0, n_1, \ldots, n_{m+1}]$, the class of FFNN functions $\Psi_{S,\sigma}$ with σ the standard sigmoid activation is a subclass of functions of $H^1(\Omega) \cap L^\infty(\Omega)$ as Ω is bounded. We define the neural network penalized PB functional $J_{\Gamma,\lambda} : \Theta_S \to \mathbb{R}$ by

$$J_{\Gamma,\lambda}[\theta] = I_{\Gamma,\lambda}[\psi_\theta] = \int_\Omega \left[\frac{\varepsilon_\Gamma}{2}|\nabla\psi_\theta|^2 - f\psi_\theta + \chi_+ B(\psi_\theta)\right] dx + \lambda \int_{\partial\Omega} (\psi_\theta - g)^2 \, dS \quad \forall \theta \in \Theta_S. \tag{3.3}$$

Note that the electrostatic energy is approximated by $\sup_{\theta\in\Theta_S}(-J_{\Gamma,\lambda}[\theta])$ with a large $\lambda > 0$; cf. (2.2) and (2.3).

We remark that $H^1(\Omega)$ is a vector space, and hence convex set, of functions and the original penalized functional $I_{\Gamma,\lambda} : H^1(\Omega) \to \mathbb{R} \cup \{+\infty\}$ is a convex functional. However, the functional $J_{\Gamma,\lambda} : \Theta_S \to \mathbb{R}$ may not be convex on Θ_S. First, the functional $J_{\Gamma,\lambda}$ is equivalently defined on $\Psi_{S,\sigma}$, and the set of neural network functions $\Psi_{S,\sigma}$ may not be a convex set. For instance, with the simplest possible architecture $S = [1, 1, 1]$, both $\psi_1(x) = \sigma(x)$ and $\psi_2(x) = \sigma(-2x)$ are functions in $\Psi_{S,\sigma}$ but $\psi(x) = (\psi_1(x) + \psi_2(x))/2$ is not in $\Psi_{S,\sigma}$. This is because any function in $\Psi_{S,\sigma}$ with σ the standard sigmoid function must be a monotonic function while ψ is not as $\psi'(0) = -1/4 < 0$ and $\psi'(\pm\infty) = 0$. By scaling, this argument applies to treating $\Psi_{S,\sigma}$ as a class of functions defined on a finite interval. Second, a convex, even linear, functional on $\Psi_{S,\sigma}$ may not be a convex function on Θ_S. For example, with $S = [1, 1, 1]$ and σ the sigmoid function, the functional $F : \Psi_{S,\sigma} \to \mathbb{R}$, defined by

$$F[\theta] = \int_0^1 \psi'_\theta(x)dx \qquad \forall \theta \in \Theta_S,$$

is not convex. This is because generally $\theta = [a, b, c, d]$ $(a, b, c, d \in \mathbb{R})$, $\psi_\theta(x) = c\sigma(ax + b) + d$, and $F[\theta] = c[\sigma(a + b) - \sigma(b)] =: f(a, b, c, d)$. Setting $b = 0$ and $c = 1$, we see that $f(a, 0, 1, d) = \sigma(a) - 1/2$ is not convex in a.

A Monte Carlo approximation of the integral $J_{\Gamma,\lambda}[\theta]$ is given by $\hat{J}_{\Gamma,\lambda}[\theta] \approx J_{\Gamma,\lambda}[\theta]$, where

$$\begin{aligned}\hat{J}_{\Gamma,\lambda}[\theta] = \frac{\mathrm{vol}(\Omega)}{N_{\Omega,\mathrm{total}}} &\left[\sum_{i=1}^{N_{\Omega,\mathrm{total}}} \left(\frac{\varepsilon_\Gamma(x_i)}{2}|\nabla\psi_\theta(x_i)|^2 - f\psi_\theta(x_i)\right) + \sum_{i=1, x_i\in\Omega_+}^{N_{\Omega,\mathrm{total}}} B(\psi_\theta(x_i))\right] \\ &+ \lambda \frac{\mathrm{area}(\partial\Omega)}{N_{\partial\Omega,\mathrm{total}}} \sum_{j=1}^{N_{\partial\Omega,\mathrm{total}}} [\psi_\theta(y_j) - g(y_j)]^2 \qquad \forall \theta \in \Theta_S.\end{aligned} \tag{3.4}$$

Here, $N_{\Omega,\mathrm{total}}$ and $N_{\partial\Omega,\mathrm{total}}$ are positive integers, $x_1, \ldots, x_{N_{\Omega,\mathrm{total}}} \in \Omega$ and $y_1, \ldots, y_{N_{\partial\Omega,total}} \in \partial\Omega$ are independently and uniformly sampled, and vol(Ω) and area$(\partial\Omega)$ are the volume and surface area of the region Ω and its boundary $\partial\Omega$, respectively. We call $\hat{J}_{\Gamma,\lambda}$ a neural network loss function. Note that the discontinuity presented in the inhomogeneous dielectric coefficient ε_Γ is treated naturally in the Monte Carlo approximation. Thus, a neural network approach may be advantageous in solving problems with complex geometries.

To apply the SGD algorithm to minimize the loss function, we need to calculate the gradient of the loss function with respect to the network weights. With a fixed ordering, each set of network weights $\theta = (W_i, b_i)_{i=1}^{m+1} \in \Theta_S$ can be identified as a vector of dimension $\sum_{i=1}^{m+1}(n_{i-1} + 1)n_i$. If we denote by θ_k the kth component of this long vector, then

$$\partial_{\theta_k} \hat{J}_{\Gamma,\lambda}[\theta] = \frac{\mathrm{vol}(\Omega)}{N_{\Omega,\mathrm{total}}} \Bigg[\sum_{i=1}^{N_{\Omega,\mathrm{total}}} \left(\varepsilon_\Gamma(x_i) \nabla \psi_\theta(x_i) \cdot \nabla \partial_{\theta_k} \psi_\theta(x_i) - f \partial_{\theta_k} \psi_\theta(x_i) \right) \\ + \sum_{i=1, x_i \in \Omega_+}^{N_{\Omega,\mathrm{total}}} B'(\psi_\theta(x_i)) \partial_{\theta_k} \psi_\theta(x_i) \Bigg] + 2\lambda \frac{\mathrm{area}(\partial\Omega)}{N_{\partial\Omega,\mathrm{total}}} \sum_{j=1}^{N_{\partial\Omega,\mathrm{total}}} \left[\psi_\theta(y_j) - g(y_j) \right] \partial_{\theta_k} \psi_\theta(y_j),$$

where $\partial_{\theta_k}\psi_\theta$ for all k are computed using the back-propagation [5, 17, 35]. The gradient (or steepest) descent updates $\theta^{(n)}$ to $\theta^{(n+1)}$ by $\theta_k^{(n+1)} = \theta_k^{(n)} - \eta \partial_{\theta_k} \hat{J}_{\Gamma,\lambda}[\theta^{(n)}]$ for all k, where η denotes the learning rate, i.e., the step size in a step of steepest descent.

Numerical algorithm. We use the SGD algorithm ADAM [20] to minimize the loss function (3.4). This is an iteration with two loops, an outer (or global) and inner (or local) loop with the total number of steps N_{global} and N_{local}, respectively. An initial set of weights $\theta^{(0)} \in \Theta_S$ are generated randomly with a multi-variate Gaussian distribution. For the kth global step ($1 \le k \le N_{\mathrm{global}}$), we select N_Ω sample points $x_i \in \Omega$ and $N_{\partial\Omega}$ sample points $y_j \in \partial\Omega$ uniformly at random and independently, and denote by $\hat{J}_{\Gamma,\lambda}^{(k)}[\theta]$ the sum of those corresponding terms in the loss function $\hat{J}_{\Gamma,\lambda}[\theta]$. The expression of $\hat{J}_{\Gamma,\lambda}^{(k)}[\theta]$ is the same as that of $\hat{J}_{\Gamma,\lambda}[\theta]$, with $N_{\Omega,\mathrm{total}}$ and $N_{\partial\Omega,\mathrm{total}}$ replaced by N_Ω and $N_{\partial\Omega}$, respectively. Here, N_Ω and $N_{\partial\Omega}$ are two pre-selected and fixed positive numbers. All the numbers N_Ω, $N_{\partial\Omega}$, $N_{\Omega,\mathrm{total}}$, $N_{\partial\Omega,\mathrm{total}}$, and N_{global} are so chosen such that $N_{\Omega,\mathrm{total}}/N_\Omega = N_{\partial\Omega,\mathrm{total}}/N_{\partial\Omega} = N_{\mathrm{global}}$. The number $N_\Omega + N_{\partial\Omega}$ is the batch size.

We minimize $\hat{J}_{\Gamma,\lambda}^{(k)}[\theta]$ using the ADAM optimizer [20], which is another iteration through the inner loop with N_{local} steps using the trained weights from minimizing $\hat{J}_{\Gamma,\lambda}^{(k-1)}[\theta]$ if $k \ge 2$ or using $\theta^{(0)}$ if $k = 1$ as the initial weights. Note that N_{local} is the number of epochs. Note also that the total number N_{iter} of overall iteration steps is $N_{\mathrm{iter}} = N_{\mathrm{global}} \cdot N_{\mathrm{local}}$. In our implementation, we do not generate all the $N_{\Omega,\mathrm{total}} + N_{\partial\Omega,\mathrm{total}}$ sample points at once. Rather, we generate $N_\Omega + N_{\partial\Omega}$ sample points in each global step. Our algorithm is summarized in Algorithm 1.

4 Numerical Tests

We construct a radially symmetric model system of a dielectric-boundary PBE and boundary value for which we have the exact solution. With such a model system, we conduct our neural network simulations for minimizing the corresponding penalized PB energy functional (2.3) in a three-dimensional setting. After testing the convergence of Monte Carlo integration, we numerically examine the validity of our penalty model, test the convergence of our algorithm with respect to the learning rate, batch size, and network architecture, and study the growth of network weights during the iteration of stochastic optimization.

A model system and simulation set up. We set $\Omega = (-L, L)^3$ for some $L > 0$, $\Gamma = \{x \in$

```
Algorithm 1 Neural network method for minimizing the PB energy functional
  Input
    • Model parameters: $\Omega$, $\Gamma$, $\varepsilon_-$, $\varepsilon_+$, the function $B$, $f$, $g$, and the penalty coefficient $\lambda$.
    • Neural network hyper-parameters: architecture $S$, activation function $\sigma$, learning rate $\eta$,
      number of sample points $N_\Omega$ and $N_{\partial\Omega}$, and total number of steps $N_{\text{global}}$ and $N_{\text{local}}$.
  Initialization
    • Initialize all the neural network weights.
  for $k = 1$ to $N_{\text{global}}$ do
    • Generate $N_\Omega$ random sample points $x_1, \dots, x_{N_\Omega} \in \Omega$ and $N_{\partial\Omega}$ random sample points
      $y_1, \dots, y_{N_{\partial\Omega}} \in \partial\Omega$, all uniformly and independently.
    • Formulate $\hat{J}^{(k)}_{\Gamma,\lambda}[\theta]$, the part of the loss function using the sampled points.
    for $j = 1$ to $N_{\text{local}}$ do
      • Compute the gradient $\nabla_\theta \hat{J}^{(k)}_{\Gamma,\lambda}$.
      • Use the ADAM optimizer to minimize $\hat{J}^{(k)}_{\Gamma,\lambda}$ and update the weights $\theta$.
    end for
  end for
  Output: Electrostatic potential and the PB energy.
```

$\mathbb{R}^3 : |x| = R\}$ for some $R \in (0, L)$, $\Omega_- = \{x \in \mathbb{R}^3 : |x| < R\}$, and $\Omega_+ = \Omega \setminus (\Gamma \cup \Omega_-)$, We define

$$
\begin{aligned}
B(s) &= \cosh(s) - 1 \quad \text{if } s \in \mathbb{R}, \\
f(x) &= \begin{cases} f_0 & \text{if } x \in \Omega_-, \\ \sinh\left(\dfrac{f_0 R^3 \exp(\alpha R)}{3\varepsilon_+(\alpha R + 1)} \cdot \dfrac{\exp(-\alpha |x|)}{|x|} \right) - \dfrac{f_0 R^3 \exp(\alpha R)}{3\varepsilon_+(\alpha R + 1)} \cdot \dfrac{\exp(-\alpha|x|)}{|x|} & \text{if } x \in \Omega_+, \end{cases} \\
g(x) &= \frac{f_0 R^3 \exp(\alpha R)}{3\varepsilon_+(\alpha R + 1)} \cdot \frac{\exp(-\alpha|x|)}{|x|} \quad \text{if } x \in \partial\Omega, \\
\phi_\Gamma(x) &= \begin{cases} -\dfrac{f_0}{6\varepsilon_-}|x|^2 + \dfrac{f_0}{6\varepsilon_-}R^2 + \dfrac{f_0}{3\varepsilon_+(\alpha R + 1)} R^2 & \text{if } x \in \Omega_-, \\ \dfrac{f_0 R^3 \exp(\alpha R)}{3\varepsilon_+(\alpha R + 1)} \cdot \dfrac{\exp(-\alpha|x|)}{|x|} & \text{if } x \in \Omega_+. \end{cases}
\end{aligned}
$$

Here, ε_- and ε_+ are two distinct positive numbers, $\alpha = 1/\sqrt{\varepsilon_+}$, and $f_0 \in \mathbb{R}$ is a constant. We verify that ϕ_Γ is a solution to the dielectric PBE (1.1) with the boundary condition $\phi_\Gamma = g$ on $\partial\Omega$. Equivalently, ϕ_Γ minimizes the PB energy functional (2.1) among all functions ϕ that satisfy the boundary condition $\phi = g$ on $\partial\Omega$.

We set $L = 1$, $R = 0.75$, $\varepsilon_- = 1$, $\varepsilon_+ = 80$, and $f_0 = 10$, and minimizing the loss function (3.4) using our neural network algorithm. Unless otherwise stated, the maximum number of local steps is $N_{\text{local}} = 10$. By numerical integration based on a finite-difference grid of grid size $h = 0.005$, we obtain the minimum energy $I_\Gamma[\phi_\Gamma] \approx -3.6172$, where I_Γ is the PB energy functional (2.1). We define the relative error Err-P of the electrostatic potential and Err-E of the minimum energy to be

$$
\text{Err-P} = \frac{\|\Phi - \phi_\Gamma\|_2}{\|\phi_\Gamma\|_2} \quad \text{and} \quad \text{Err-E} = \frac{|I_\Gamma[\Phi] - I_\Gamma[\phi_\Gamma]|}{|I_\Gamma[\phi_\Gamma]|}, \tag{4.1}
$$

respectively, where Φ is a simulated neural network approximation of the potential ϕ_Γ.

Test on the convergence of Monte Carlo integration. We use the numerical integration method to compute the following volume integral (also called the interior integral here) and the boundary integral

$$\int_\Omega \left[\frac{\varepsilon_\Gamma}{2} |\nabla \phi_\Gamma|^2 - f\phi + \chi_+ B(\phi_\Gamma) \right] dx \quad \text{and} \quad \int_{\partial\Omega} \phi_\Gamma^2 \, dS,$$

respectively, with finite-difference grids of size $h = 0.005$ for both integrals. These integral values are used as "exact" values. We then use the Monte Carlo integration method with different number of sample points to approximate these integrals and calculate the absolute errors. In Figure 2, we plot in the log-log scale the absolute error vs. the number of sampling points for the interior integral (left) and the boundary integral (right). We observe the convergence rate $O(N^{-1/2})$ as expected. Note that this convergence rate is independent of the dimension of integral region, three or two dimension. Moreover, for the volume integral, the rate is not deteriorated by the discontinuity of the integrand from the term ε_Γ. This is true if we replace the constructed ϕ_Γ by the minimizer of the functional $I_{\Gamma,\lambda}$ defined in (2.3) for a general setting as in Theorem 2.1, Such a minimizer is piecewise smooth and bounded (cf. [25]) and consequently the square of the integrand in the volume integral is integrable, satisfying the finite-variance requirement in achieving the desired convergence of Monte Carlo integration. Similarly, the convergence rate is not affected by the interfacial discontinuity presented in ε_Γ if ϕ_Γ is replaced by any neural-network function, as such a function and its gradient are bounded.

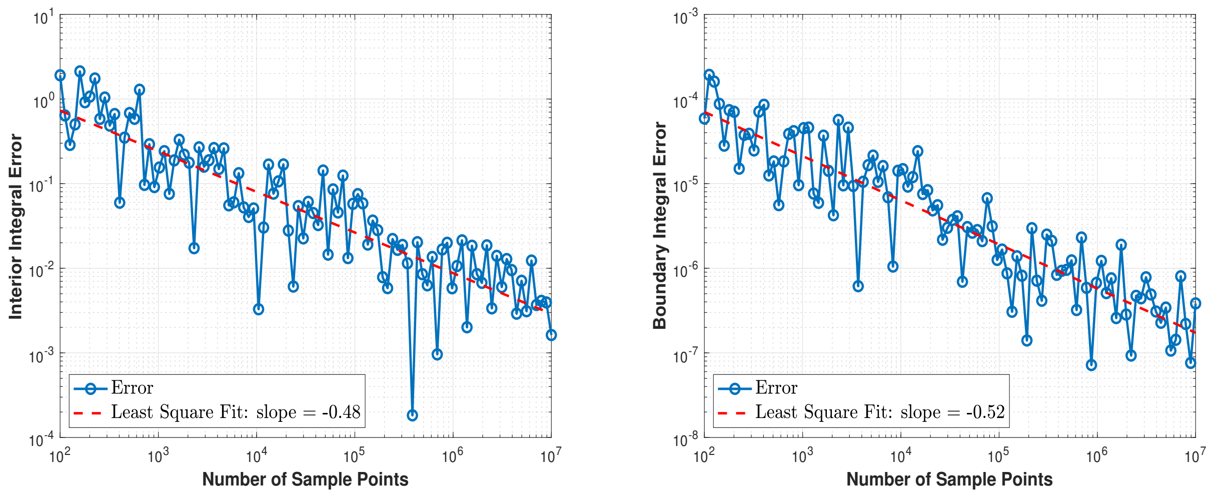

Figure 2: Log-log plot of the absolute error vs. the number of sample points of the Monte Carlo simulation of the interior (left) and the boundary (right) integral. The slope is approximately $-1/2$ for both cases, indicating the convergence rate of 1/2 as expected.

Test on the validity of the penalty method. We compare the errors Err-E and Err-P defined in (4.1) with different penalty coefficients $\lambda = 25, 100$, and 250 with the network architecture $S = [3, 30, 20, 15, 10, 1]$, batch size = 6144 with $N_\Omega = N_{\partial\Omega} = 3072$), and learning rate = 10^{-2}. Note that for numerical tests, we choose $N_\Omega = N_{\partial\Omega}$ for simplicity of implementation, as the convergence rate of the Monte Carlo integration is independent of dimension. Our numerical simulation results are presented in Table 1 and Figure 3. We observe that, with the same λ, both Err-E and Err-P decrease as the total number of steps N_{iter} increases. With the same N_{iter} and as the value of λ increases, the relative error Err-P decreases while the relative error Err-E does not decrease. This may be due to a higher initial energy value of the penalty term with a larger value of λ, requiring more iteration steps to reduce the total energy. In fact, all the relative errors Err-E for $\lambda = 25$ and $N_{\text{iter}} = 100,000$, $\lambda = 100$ and $N_{\text{iter}} = 200,000$, and $\lambda = 250$ and $N_{\text{iter}} = 300,000$ are very close to each other. It is interesting to see that the sum of Err-E and Err-P is roughly independent of λ. Overall, the convergence with respect to the increase of the penalty coefficient is achieved, which validates our penalty method. Moreover, while a large value of λ is required by the convergence of the penalty model (cf. Theorem 2.2), our tests show that a moderate value of λ performs well.

Total Number of Steps N_{iter}	$\lambda = 25$		$\lambda = 100$		$\lambda = 250$	
	Err-E	Err-P	Err-E	Err-P	Err-E	Err-P
100,000	4.68%	7.36%	5.61%	5.51%	8.68%	4.39%
200,000	3.44%	7.38%	5.33%	3.50%	5.38%	2.89%
300,000	1.68%	6.88%	3.80%	2.34%	4.18%	1.96%

Table 1: Results of neural network minimization of the three-dimensional penalized PB energy functional with different penalty coefficient λ and total number of steps N_{iter}.

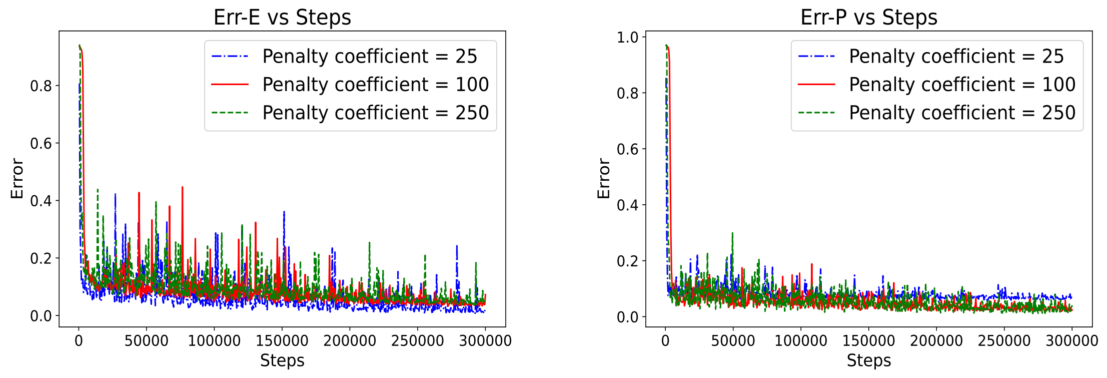

Figure 3: Err-E (left) and Err-P (right) vs. the overall iteration step in our neural network optimization of the three-dimensional PB functional with the penalty coefficient $\lambda = 25$, 100, and 250, respectively.

Convergence test: Effects of learning rate, batch size, and network architecture. We fix the penalty coefficient to be $\lambda = 250$ and the network architecture $S = [3, 30, 20, 15, 10, 1]$. We run our neural network simulations with $N_{\text{iter}} = 300,000$, $N_{\text{local}} = 10$, the batch size 1536 ($N_{\Omega} = N_{\partial\Omega} = 768$), or 3072 ($N_{\Omega} = N_{\partial\Omega} = 1536$), or 6144 ($N_{\Omega} = N_{\partial\Omega} = 3072$), and the learning rate 10^{-2} or 10^{-3}. All the simulation results are summarized in Table 2. We observe the convergence in each case with a large or small learning rate and a large or small batch size. Moreover, for a fixed batch size, our neural network simulation with a larger learning rate leads to faster convergence and smaller errors. Note that a very large learning rate may lead to unstable iterations. Similarly, for a fixed learning rate, simulations with a larger batch size perform better than those with a smaller batch size. This is because a large batch size means more sample points for Monte Carlo integration.

Learning Rate	Batch size	Err-E	Err-P
1e-03	1536	7.24%	5.35%
	3072	6.18%	4.16%
	6144	6.35%	3.56%
1e-02	1536	7.08%	4.46%
	3072	5.79%	3.68%
	6144	4.18%	1.96%

Table 2: Relative errors in the neural network minimization of the penalized PB energy functional in the three-dimensional setting with different learning rates and batch sizes.

In Figure 4, we plot the neural network penalized PB energy and the relative L^2-error of the electrostatic potential ϕ vs. the overall iteration step in the optimization process. It is clear that all the simulations converge despite some oscillations. As before, we observe that simulations with relatively larger learning rates and larger batch sizes perform better. In addition, we also observe that more oscillations appear in iteration with a larger learning rate.

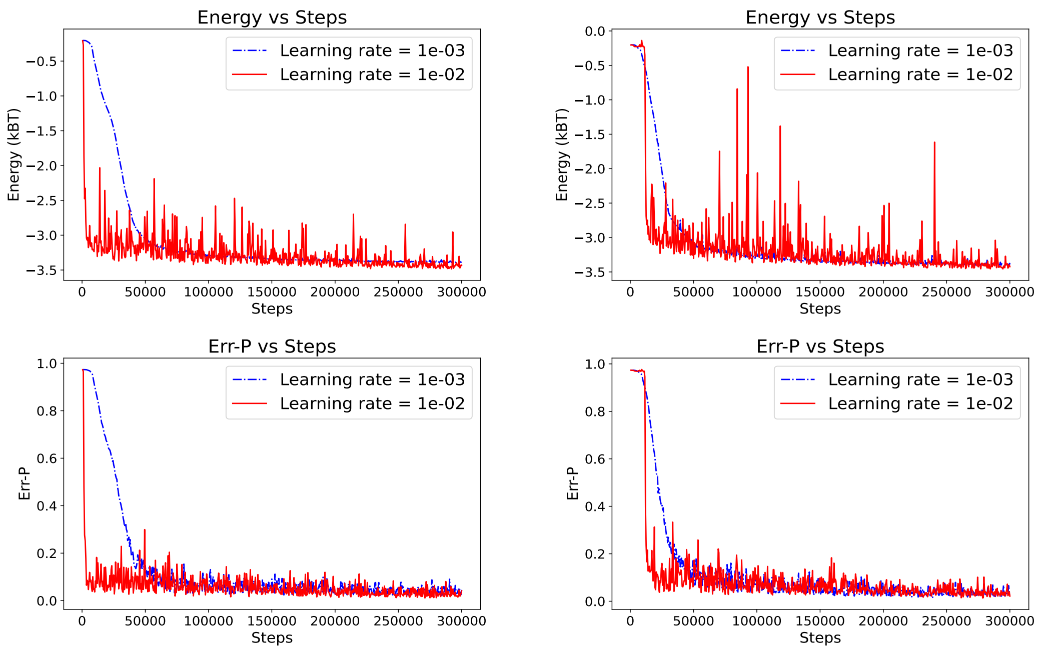

Figure 4: The neural network optimization of the penalized PB energy (top) and the relative error L^2-Err-P of the electrostatic potential ϕ (bottom) vs. the overall iteration step. The batch sizes are 6144 (left) and 3072 (right), respectively.

Finally, we fix the penalty coefficient $\lambda = 250$ and the batch size 6144 ($N_\Omega = N_{\partial\Omega} = 3072$) to test the effect of learning rate and network architecture on the convergence of our algorithm. We use our neural network algorithm to minimize the loss function that is constructed from the penalized PB energy functional for a shallow network $[3, 30, 15, 1]$ and a deep network $[3, 20, 10, 10, 5, 1]$ with a learning rate 10^{-3} or 10^{-2}. Note that these networks have the same total number of neurons. We show in Table 3 the relative error of the simulated energy and the relative L^2-error of the simulated electrostatic potential. In Figure 5, we plot the neural network energy relaxation and relative error of the electrostatic potential vs. the overall iteration step. We see clearly from this table and these plots that with the same learning rate, a deep network performs better than a shallow one in terms of convergence.

Network Architecture	Learning Rate	Err-E	Err-P
[3, 30, 15, 1]	1e-03	13.33%	9.71%
	1e-02	6.76%	3.57%
[3, 20, 10, 10, 5, 1]	1e-03	5.27%	3.62%
	1e-02	5.11%	3.83%

Table 3: Relative errors of the neural network simulated electrostatic energy and potential with two different network architectures and two different learning rates.

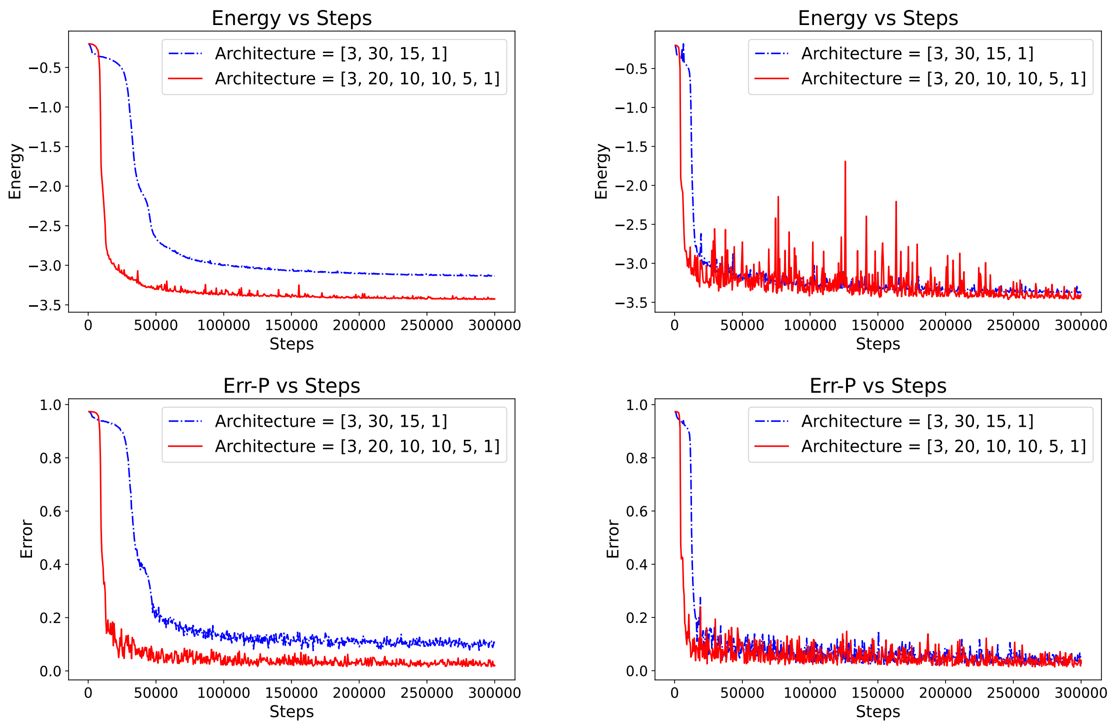

Figure 5: The neural network optimization of the penalized PB energy (top) and the relative L_2-error Err-P of the electrostatic potential ϕ (bottom) vs. the overall iteration step for learning rate 10^{-3} (left) and 10^{-2} (right) for two different network architectures.

Test on the growth of weights. For a network architecture $S = [n_0, n_1, \ldots, n_{m+1}]$. We define the size of the set of weights $\theta = (W_i, b_i)_{i=1}^{m+1} \in \Theta_S$ (cf. (3.1)) by

$$\|\theta\| = \|(W_i, b_i)_{i=1}^{m+1}\| = \sum_{i=1}^{m+1} \left(\|W_i\|_F^2 + \|b_i\|_2^2 \right),$$

where $\|\cdot\|_F$ and $\|\cdot\|_2$ are the matrix Frobenius norm and vector l^2-norm, respectively. The theory of neural network approximations indicates that, in the case of non-attainment of best approximation of a given function, the size of weights of neural networks that are closer and closer to the function can grow unbounded [30, 33]. Here we test such growth in our neural network optimization of the penalized PB energy functional. Figure 6 shows the size of network weights vs. the iteration step in our neural network SGD iteration for two learning rates. We observe that the weights do not grow with a small learning rate but grow largely with a large learning rate.

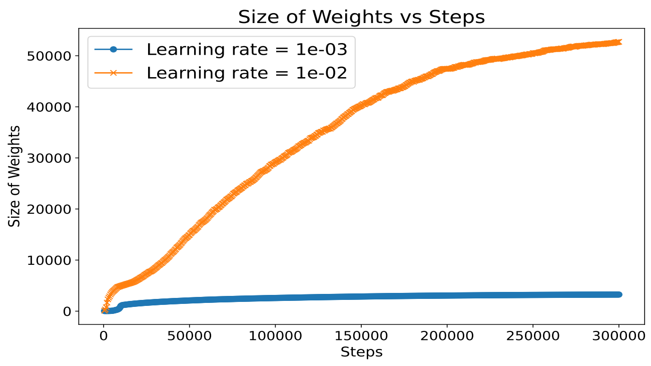

Figure 6: The size of weights vs. the overall iteration step in the neural network stochastic minimization of the penalized PB energy functional for two different learning rates.

5 Application to Biomolecular Solvation

We apply our neural network approach to the dielectric-boundary PBE in conjunction with a variational implicit-solvent model (VISM) of charged molecules immersed in aqueous solvent; cf. Figure 1 [8, 11, 12, 47]. Central in this model is an effective free-energy functional $F[\Gamma]$ of all possible solute-solvent interfaces (i.e., dielectric boundaries) Γ, given by

$$F[\Gamma] = P_0 \mathrm{Vol}(\Omega_-) + \gamma_0 \int_\Gamma (1 - 2\tau H)\, dS + \rho_{\mathrm{w}} \int_{\Omega_+} U(X, x)\, dx + E_{\mathrm{ele}}[\Gamma]. \tag{5.1}$$

Here, Ω_- and Ω_+ are the solute molecular region and solvent region, respectively, P_0 is the difference between the pressure outside and inside the molecular region Ω_-, γ_0 is the constant surface tension, τ is an adjustable parameter (called the Tolman length), H is the mean curvature, ρ_{w} is the bulk solvent density, $U(X, x) = \sum_{i=1}^N U_{\mathrm{LJ}}^{(i)}(|x - x_i|)$ with each $U_{\mathrm{LJ}}^{(i)}$ a Lennard-Jones (LJ) potential, and all $x_1, \dots, x_N$ are the solute atoms. For each i, the LJ potential is given by $U_{\mathrm{LJ}}^{(i)}(r) = 4\varepsilon_i[(\sigma_i/r)^{12} - (\sigma_i/r)^6]$ with σ_i and ε_i given parameters. The last part $E_{\mathrm{ele}}[\Gamma]$ is the electrostatic energy and it is given in (2.2), where ϕ_Γ is the unique solution to the boundary-value problem of the dielectric-boundary PBE (1.1). Table 4 collects the typical values of parameters in the model that are used in our simulations. The LJ parameters will be specified later.

Parameters	Descriptions	Estimated Values	Units
T	temperature	300	Kelvin
P_0	pressure difference	0	bar
γ_0	constant surface tension	0.1315	$k_{\mathrm{B}}T/$ Å^2
τ	Tolman length	0.76	Å
ρ_{w}	bulk solvent density	0.0331	Å^{-3}
ε_-	relative dielectric permittivity in Ω_-	1	ε_0
ε_+	relative dielectric permittivity in Ω_+	78	ε_0

Table 4: Model parameters, where ε_0 is the vacuum permittivity [19, 42, 47].

5.1 Solvation of a single ion

We consider a single ion modeled as a sphere of radius R with a charge of value Q placed at its center that is assumed to be the origin. This system is spherically symmetric. With our previous notations (cf. Figure 1), we have $\Omega_- = \{x \in \mathbb{R}^3 : |x| < R\}$, $\Omega_+ = \{x \in \mathbb{R}^3 : |x| > R\}$, $\Gamma = \{x \in \mathbb{R}^3 : |x| = R\}$ for some $R > 0$. The dielectric coefficient $\varepsilon_\Gamma(x) = \varepsilon_R(|x|)$ is given by $\varepsilon_R(|x|) = \varepsilon_-$ if $|x| < R$ and $\varepsilon_R(|x|) = \varepsilon_+$ if $|x| > R$. In addition, $N = 1$ and $x_1 = 0$ is the origin. The LJ parameters for the single ion are denoted by $\varepsilon_{\mathrm{LJ}} = \varepsilon_1$ and $\sigma_{\mathrm{LJ}} = \sigma_1$. The fixed charge density is now given by $f = Q\delta_0$, where δ_0 is the Dirac delta function concentrated on the origin. The VISM free-energy functional now becomes a function of R, given by [7, 42, 47]

$$F(R) = \frac{4\pi}{3}P_0R^3 + 4\pi\gamma_0R^2 - 8\pi\gamma_0\tau R + 16\pi\rho_{\mathrm{w}}\varepsilon_{\mathrm{LJ}}\left(\frac{\sigma_{\mathrm{LJ}}^{12}}{9R^9} - \frac{\sigma_{\mathrm{LJ}}^6}{3R^3}\right) + E_{\mathrm{ele}}(R), \tag{5.2}$$

where the electrostatic energy $E_{\mathrm{ele}}(R)$ is now a function of R. We shall neglect the effect of surrounding mobile ions, i.e., set $B(s) = 0$ for all $s \in \mathbb{R}$; cf. (1.1). In this case, the electrostatic energy $E_{\mathrm{ele}}(R)$ is approximated by the Born energy [4]

$$E_{\mathrm{Born}}(R) = \frac{Q}{2}\left(\phi_{\mathrm{B},R} - \phi_{\mathrm{C}}\right)(0) = \frac{Q^2}{8\pi R}\left(\frac{1}{\varepsilon_+} - \frac{1}{\varepsilon_-}\right), \tag{5.3}$$

where $\phi_{\mathrm{B},R}$, the Born potential, is defined by

$$-\nabla \cdot \varepsilon_R \nabla \phi_{\mathrm{B},R} = Q\delta_0 \quad \text{in } \mathbb{R}^3 \qquad \text{and} \qquad \phi_{\mathrm{B},R}(\infty) = 0,$$

and $\phi_{\mathrm{C}}(x) = Q/(4\pi\varepsilon_-|x|)$ is the Coulomb field. One can verify that

$$\phi_{\mathrm{B},R}(x) = \begin{cases} \dfrac{Q}{4\pi\varepsilon_-|x|} + \dfrac{Q}{4\pi R}\left(\dfrac{1}{\varepsilon_+} - \dfrac{1}{\varepsilon_-}\right) & \text{in } \Omega_-, \\ \dfrac{Q}{4\pi\varepsilon_+|x|} & \text{in } \Omega_+. \end{cases} \tag{5.4}$$

We test our neural network method for accurately calculating the electrostatic energy. To do so, we consider a bounded solvation region $\Omega = \{x \in \mathbb{R} : |x| < A\}$ for a fixed $A > 0$ with Ω_- and Γ same as above and $\Omega_+ = \{x \in \mathbb{R}^3 : R < |x| < A\}$. We approximate the electrostatic energy $E_{\mathrm{ele}}(R)$ by

$$E_{\mathrm{ele},A}(R) = \frac{Q}{2}\left(\phi_{R,A} - \phi_{\mathrm{C}}\right)(0),$$

where $\phi_{R,A}$ is the unique solution to the boundary-value problem of Poisson's equation

$$-\nabla \cdot \varepsilon_R \nabla \phi_{R,A} = Q\delta_0 \quad \text{in } \Omega \qquad \text{and} \qquad \phi_{R,A}(x) = g \quad \text{if } |x| = A, \tag{5.5}$$

where g is a constant. We can verify that

$$\phi_{R,A}(x) = \begin{cases} \dfrac{Q}{4\pi\varepsilon_-|x|} + \dfrac{Q}{4\pi R}\left(\dfrac{1}{\varepsilon_+} - \dfrac{1}{\varepsilon_-}\right) - \dfrac{Q}{4\pi\varepsilon_+ A} + g & \text{in } \Omega_-, \\ \dfrac{Q}{4\pi\varepsilon_+|x|} - \dfrac{Q}{4\pi\varepsilon_+ A} + g & \text{in } \Omega_+, \end{cases} \tag{5.6}$$

$$E_{\text{ele},A}(R) = \frac{Q}{2}\left(\phi_{R,A} - \phi_{\text{C}}\right)(0) = \frac{Q^2}{8\pi R}\left(\frac{1}{\varepsilon_+} - \frac{1}{\varepsilon_-}\right) - \frac{Q^2}{8\pi\varepsilon_+ A} + \frac{Qg}{2}. \tag{5.7}$$

We also reformulate this boundary-value problem with a point charge to an energy-minimization problem (cf. Lemma 3.1 in [25] and [19])

$$E_{\text{ele},A}(R) = -\min_{\phi \in H^1_g(0,A)} I_R[\phi] = \max_{\phi \in H^1_{\omega,g}(0,A)} (-I_R[\phi]), \tag{5.8}$$

where

$$I_R[\phi] = 4\pi \int_0^A \frac{\varepsilon_R(r)}{2}\left[(\phi - \phi_{\text{B},R})'(r)\right]^2 r^2\, dr - E_{\text{ele},A}(R),$$

$$H^1_g(0,A) = \{\phi : (0,A) \to \mathbb{R} \text{ weakly differentiable: } \phi(A) = g \text{ and } \int_0^A (\phi^2 + \phi'^2) r^2 dr < \infty\}.$$

The reference field $\phi_{\text{B},R}$ in the integral defining $I_R[\phi]$ is the Born potential (5.4). Note that by (5.6) and (5.4) that $\phi_{R,A} \in H^1_g(0,A)$ and $\phi_{R,A} = \phi_{\text{B},R} + g - Q/(4\pi\varepsilon_+ A)$. Thus $\phi_{R,A}$ is the unique minimizer of I_R over $H^1_g(0,A)$ and hence it satisfies the corresponding Euler–Lagrange equation (5.5). Moreover, the minimum is indeed given by $-E_{\text{ele},A}$ in (5.8).

Let $\lambda > 0$. Note that for $\phi \in H^1_g(0,A)$ the function $u = \phi - \phi_{\text{B},R} \in H^1(0,A)$ has the boundary value $u = g - Q/(4\pi\varepsilon_+ A)$ on the boundary $|x| = A$. We thus define the penalized energy functional $J_{R,\lambda} : H^1(0,A) \to \mathbb{R}$ by

$$J_{R,\lambda}[u] = 4\pi \int_0^A \frac{\varepsilon_R(r)}{2}[u'(r)]^2 r^2\, dr + 4\pi A^2 \lambda \left[u(A) - g + \frac{Q}{4\pi\varepsilon_+ A}\right]^2 - E_{\text{ele},A}(R). \tag{5.9}$$

Note that the electrostatic energy is approximated by $\max_{u \in H^1(0,A)}(-J_{R,\lambda}[u])$ for large $\lambda > 0$. We use our neural network method to minimize this functional with a given R and a large $\lambda > 0$. Note that the penalized functional (5.9) is minimized uniquely by $u = \phi_{R,A} - \phi_{\text{B},R}$ and this exact solution is used for validating our method. In our numerical simulations, we set $g = Q/(4\pi\varepsilon_+ A)$, $A = 4$, and $\lambda = 250$. In this case, we have $\phi_{R,A} = \phi_{\text{B},R}$ exactly; cf. (5.4) and (5.6).

We first consider an artificial cation with varying charge value Q. Due to the charge symmetry in a continuum model, an anion is the same in our simulations. All model parameters in the VISM energy functionals (5.1) and (5.2), except the LJ parameters, are summarized in Table 4. We set the only LJ parameters to be $\sigma_{\text{LJ}} = 3.5$ Å and $\varepsilon_{\text{LJ}} = 0.3\ k_{\text{B}}T$. For our simulations, the neural network architecture is $[1, 20, 20, 20, 20, 1]$, the batch size is 1600 with $N_\Omega = 1536$ and $N_{\partial\Omega} = 64$. Here and below, we choose N_Ω and $N_{\partial\Omega}$ so that $\sqrt[3]{N_\Omega} \approx \sqrt{N_{\partial\Omega}}$. The number of global and local iteration steps are $N_{\text{global}} = 2,000$ and $N_{\text{local}} = 10$, respectively. Thus, the total number of iteration steps is $N_{\text{iter}} = 20,000$.

For each chosen value of Q, we get the optimal radius $R_{\min}$ that minimizes the VISM energy functional (5.2) [47]. Different Q-values and the corresponding optimal radii are displayed in the first two rows of Table 5. We then fix $R = R_{\min}$ and calculate the electrostatic energy by minimizing the loss function constructed from the penalty functional (5.9). These simulated electrostatic energy values and the exact energy values (i.e., the Born energy values (5.3)) are very close to each other, same up to the third decimal place. These common values are displayed in the third row of Table 5, marked "Electrostatic Energy". Note that the relative errors are within 10^{-5}–10^{-6} percentage. For each Q-value, and hence the value of $R = R_{\min}$, the neural network simulated electrostatic energy value is then used to replace the exact value $E_{\text{ele}}(R_{\min})$ in (5.2), providing a neural network approximation of the VISM solvation energy. These approximated solvation energy values and the exact VISM solvation energy values, given by $F(R_{\min})$ with the Born energy (5.3) for $E_{\text{ele}}(R)$ in (5.2), are the same, up to the third decimal place. These common values are listed in the last row of Table 5, marked "Solvation Energy".

Charge Q	0.0	0.5	1.0	1.5	2.0
Radius R	3.157	3.030	2.801	2.605	2.453
Electrostatic Energy	0.0	-22.685	-98.156	-237.469	-448.326
Solvation Energy	4.836	-17.413	-88.486	-216.174	-406.986

Table 5: Electrostatic and VISM solvation free energies (in k_BT) obtained by the neural network simulations are the same as the exact Born electrostatic and VISM free energies up to the third decimal place for an artificial ion with different values of the point charge Q (in e).

We now compare two methods of generating initial neural network weights. The first one is a commonly used method to generate random initial weights with some distributions. The second one is to use the trained weights from one simulation as the initial weights for the simulation of a different system. Specifically, we fix $Q = 0\,\text{e}$ and train the network using the stochastic optimization. The final weights from the optimization are then used as initial weights for the neural network simulations for the electrostatics with different Q-values. Figure 7 shows that with trained weights as initial weights, the simulation is much more efficient than untrained weights generated randomly.

We consider now the solvation free energy of real single ions K^+, Na^+, Cl^- and F^-. All the parameters are the same as above and are listed in Table 4, except those LJ parameters that are displayed in Table 6. The VISM-NN values are obtained by the same neural network simulations as above. However, in calculating the electrostatic energy for the anion Cl^- or F^-, we shift the VISM free-energy (cf. (5.2)) minimizing radius in parallel toward its center of ion by $\xi = 1$ Å and then use the shifted radius to calculate the electrostatic energy [47]. We present in Table 6 our neural network simulation results in comparison with the experimental data [31, 47]. It is clear that our neural network can accurately predict the solvation free energy for each of the ions.

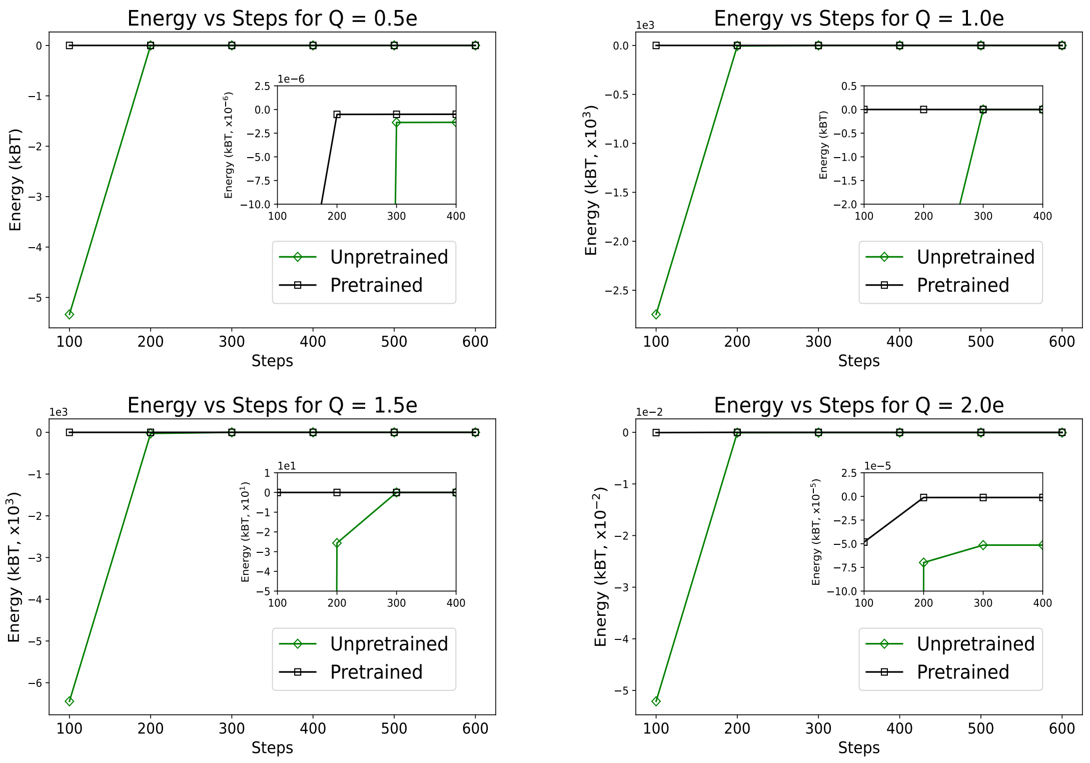

Figure 7: Profiles of the neural network approximated and shifted electrostatic energy $-J_{R,\lambda}[\psi_\theta] - E_{\text{ele,A}}(R)$ as a function of the weights θ vs. the overall iteration steps for the point charge with the different charge values Q. Note that "energy" $-J_{R,\lambda}[\psi_\theta] - E_{\text{ele,A}}(R)$ increases to 0 during the iteration. Random initial weights are marked by "Unpretrained" and initial weights being the finally trained from the simulation for $Q = 0\,e$ are marked "Pretrained."

Ions	ε (k_BT)	σ (Å)	VISM-NN	Experiment
K^+	0.008	3.85	-111.1	-117.5
Na^+	0.008	3.49	-129.9	-145.4
Cl^-	0.21	3.78	-126.1	-135.4
F^-	0.219	3.3	-171.0	-185.2

Table 6: Neural network predicted (VISM-NN) and experimental solvation free energy (in k_BT) of the single ions K^+, Na^+, Cl^- and F^- modeled as a single point charge placed at the origin.

5.2 Electrostatics of protein BphC

To assess the performance of our neural network approach to the PB electrostatics for large complex systems, we consider the protein BphC (Biphenyl-2,3-diol-1,2-dioxygenase, PBD code: 1dhy.pdb) [36], which plays an important role in molecular engineering. The sequence length of BphC is 292 so the protein has a few thousand atoms. Structurally, it consists of two domains. To examine the hydrophobic interactions in the region between two domains, we vary the separation distance d, which is defined to be that of the two geometric centers of those two domains. The structure with $d = 0$ corresponds to the native crystal structure [42, 46].

For each distance d, we numerically determine the protein surface (i.e., dielectric boundary) Γ as a local minimizer of the total VISM solvation free-energy functional (5.1) with the electrostatic energy modeled by the Coulomb-field approximation; cf. [42] for details of the model and numerical simulations. Such a minimizer is not unique in general. We are interested in two types of such a dielectric boundary minimizer. One is a single surface that wraps up all the solute atoms. We call such a configuration a dry state, as there are no solvent molecules in between the two domains of the protein. The other is a surface with possible two or more components, wrapping up tightly all the solute atoms in the two domains, respectively. In this case, we call the corresponding configuration a wet state, as there are solvent molecules in between the two domains. Such dry and wet states are known to be crucial in biomolecular processes [2, 41, 48]. Our previously developed VISM model can capture such states by relaxing the effective VISM free-energy functional (5.1) using specially designed initial surfaces. A tight initial surface, an initial surface tightly wrapping up all the protein atoms, usually leads to a wet configuration, while a loose initial surface, an initial surface loosely wrapping up all the protein atoms, usually leads to a dry configuration. We note that the variational model for generating different solvation free-energy minimizing dielectric boundaries are not implemented in many existing simulation software, making it impossible to compare our approach with those software.

Figure 8 shows a sequence of dry and wet states of the protein BphC with three different domain separation distances predicted by the VISM. With either a tight or loose initial surface, VISM predicts only one state which is dry if the separation distance is very small, e.g., $d \leq 4$ Å, while one state which is wet if d is large, e.g., $d \geq 16$ Å. For $d = 10$ Å, some differences between wet (top) and dry (bottom) states can be observed. Once the distance d is large enough, say $d = 14$ Å, the tight and loose surfaces are again very close to each other, and both configurations are wet.

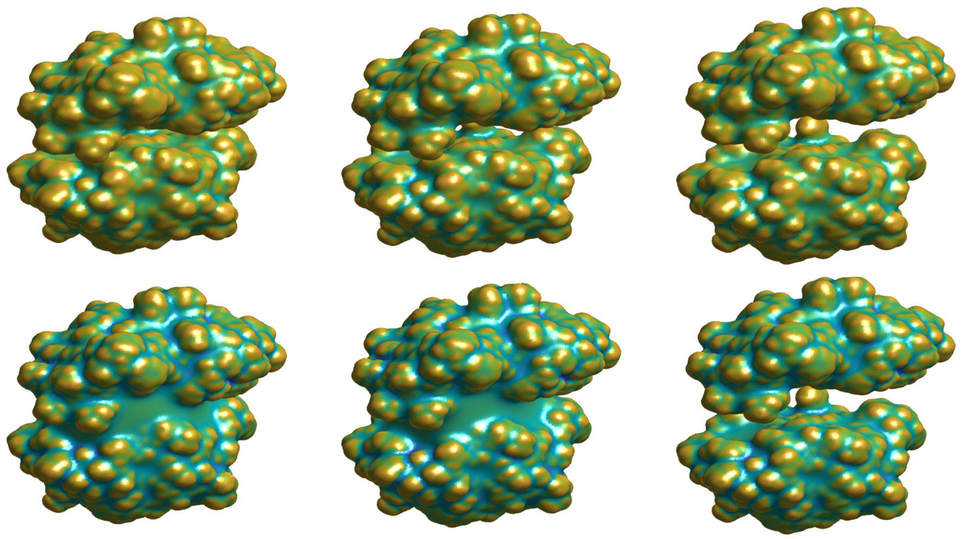

Figure 8: Protein BphC surfaces (i.e., dielectric boundaries) predicted by VISM. From left to right: Tight surfaces (Top) and loose surfaces (bottom) for $d = 10$ Å, 12 Å, and 14 Å, respectively.

We now apply our neural network approach to predicting the electrostatic free energy of the protein BphC. Let us fix a separation distance d with $4\,\text{Å} \le d \le 16\,\text{Å}$ and a tight or loose surface Γ predicted by the VISM; cf. Figure 8. We solve the PB equation (1.1) with the dielectric boundary Γ to get the electrostatic potential and also the energy, using our neural network model, i.e., minimizing the penalized functional (2.3). We assume that in the bulk there are two types of ions (i.e., $M = 2$) with bulk ionic concentrations and charges being $c_1^\infty = c_2^\infty = c^\infty = 10^{-7}\text{molar} = 10^{-7}\text{mol/L}$ and $q_1 = -q_2 = 1\,\text{e}$, respectively, where e is the elementary charge. Other modeling parameters are the same as in Table 4. The atomic partial charges of the protein are approximated by

$$f(x) = \frac{1}{(2\pi\sigma)^{3/2}} \sum_{i=1}^{N} Q_i \exp\left(-\frac{|x - x_i|^2}{2\sigma}\right),$$

where x_i and Q_i are the position and partial charges of the i-th atom of BphC with N of them, and $\sigma = 0.1$. We set the boundary value to be

$$g(x) = \sum_{i=1}^{N} \frac{Q_i \exp(-\kappa|x - x_i|)}{4\pi\varepsilon_+|x - x_i|} \qquad \forall x \in \partial\Omega,$$

where $\kappa = (\varepsilon_+ k_B \text{T} / \sum_{j=1}^{M} c_j^\infty q_j^2)^{-1/2}$ is the inverse Debye length. The penalty coefficient is chosen to be $\lambda = 1,000$. The dielectric boundary is prescribed by the zero level-set surface given by a discrete level-set function that is defined on uniform grid points. We employ an 8-layer neural network with the architecture $[3, 50, 30, 20, 15, 10, 5, 1]$. The batch size is 5376 ($N_\Omega = 4608$ and $N_{\partial\Omega} = 768$). The learning rate is set to be 1×10^{-3} and the number of global and local iteration steps are $N_{\text{global}} = 25,000$ and $N_{\text{local}} = 20$, respectively, so that total number of iteration is $N_{\text{iter}} = 500,000$.

Figure 9 shows the electrostatic energy vs. distance d for both the tight and loose surfaces from our neural network simulations. We observe that the neural network PB solutions with loose and tight dielectric boundaries exhibit the expected hysteresis phenomenon [42]. The energy profile also qualitatively matches the result from [42]. Therefore, our neural network algorithm is capable of dealing with complicated dielectric interface geometry.

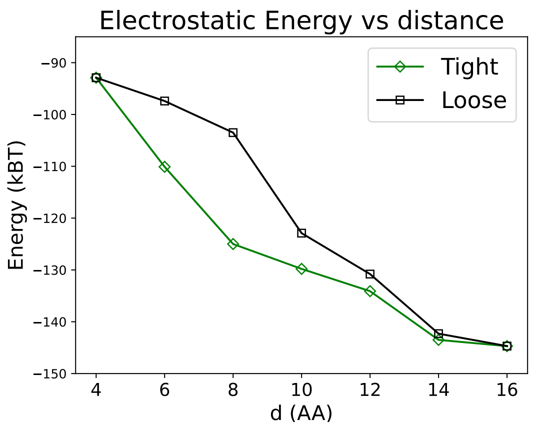

Figure 9: Neural network predictions of the electrostatic energy for a dry (marked by "Loose") or wet (marked by "Tight") state of the protein BphC at varying separation distances d.

We also explore the application of "transfer learning" to accelerate the training and optimization, as these processes often represent the most time-intensive aspect of the simulation. Specifically, we leverage the well-trained weights and biases from a fixed separation distance, and use them as initial weights and biases of neural network for the simulation for a different and nearby distance. For instance, we utilized the weights and biases trained for $d = 8$ Å to initialize the network for $d = 10$ Å in the context of a loose interface. Figure 10 shows the result for this experiment. Here, "Pretrained" is the energy profile for $d = 10$ Å with the initial weights being those finally trained from simulations for the neighboring distance $d = 8$ Å, while "Unpretrained" shows the energy profile from simulations with random initialization of weights. In these simulations, the network architecture is $S = [3, 20, 15, 10, 5, 1]$, the learning rate is 5×10^{-3}, the batch size is 15360 ($N_\Omega = 13824$ and $N_{\partial\Omega} = 1536$), and the total number of global and local iteration steps are $N_{\text{global}} = 30,000$ and $N_{\text{local}} = 10$, respectively. The total number of iteration is $N_{\text{iter}} = 300,000$. Note that the energy increases with respect to iteration steps, as this true energy is the negative of the energy we minimize; cf. Eq. (2.1). From the figure, the "Pretrained" simulations converge significantly faster than those with the "Unpretrained" setting. This highlights the potential of the transfer learning approach to significantly reduce the training time, especially in dealing with a large number of simulations for similar complex biological systems.

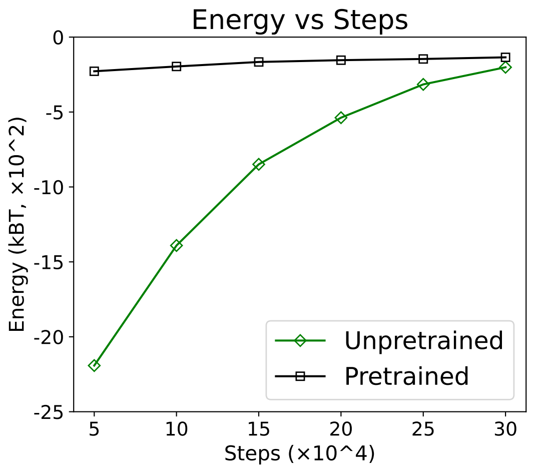

Figure 10: Electrostatic energy of BphC protein with a loose surface at domain separation $d = 10$ Å vs. the overall iteration steps. Case 1: the initial weights are randomly generated (marked "Unpretrained"). Case 2: the initial weights are those finally trained from simulations for the distance $d = 8$ Å (marked "Pretrained").

6 Conclusions

We have developed a neural network approach to solving the boundary-value problem of dielectric-boundary PBE. Solving the equation is reformulated into a variational problem of minimizing an effective PB electrostatic free-energy functional of all admissible electrostatic potentials. To enforce the boundary condition for such potentials, we introduce a penalized electrostatic free-energy functional that approximates the PB energy functional. We have proved that, as the penalty coefficient increases to infinity, the minimizer and the minimum value of the penalized functional converge to those of the PB functional, respectively. We then employ the penalized functional together with the Monte Carlo integration to construct a neural network loss function and used the SGD algorithm ADAM to minimize the loss function.

We have conducted a series of numerical tests demonstrating the convergence of our neural network algorithm with varying learning rates, batch sizes, and network architectures. The penalty model is numerically validated. In addition, we have tested the growth of the magnitude of neural network weights during the optimization and found no explosion of the weights.

Further, we have combined our neural network approach with the variational implicit-solvent model to predict the solvation free energy of some single ions, and the potential of mean electrostatic force of the protein BphC with respect to the separation distance of its two domains. Our extensive simulations indicate that the neural network can effectively handle complex geometries and predict qualitatively accurately the electrostatic potential and energy. In particular, we have reproduced the hysteresis loop for the electrostatic interaction of the protein BphC in dry and wet states with respect to its two-domain separation.

Through our simulations, we find that the learning rate, batch size, and network architecture can all significantly affect the convergence of the neural network SGD algorithm. A relatively large learning rate leads to a faster convergence but also more fluctuations. A large batch size can lead to a more accurate result but can also be less efficient. With the same number of nodes, a deep network performs generally better than a shallow network. With a fixed network architecture, a large learning rate may lead to the growth of the size of network weights in the training process.

Our most surprising and significant finding is the transferability of network weights, i.e., trained weights from simulations for one setting (e.g., with one separation distance) can be used for simulations for a different setting (e.g., a different separation distance). Such transferability is found in simulations of both a single ion and the complex protein BphC. Another significant and interesting finding is that networks approximately minimizing the loss function can be far away from each other as their sizes can be very different, indicating possibly that the patch of the loss surface with near minimum loss value is a large and flatten region.

We now discuss several issues and suggest possible improvements for future work.

(1) *Solving PDEs: machine learning vs. traditional methods.* In comparison with traditional methods, such as finite element and finite difference methods, for solving (low-dimensional) PDEs, our neural network simulations are not evidently more accurate or efficient in general. A loss function is often highly nonconvex and constructed using random samples. Training neural networks to minimizing the loss function usually takes long time. In our study, we find that the convergence in minimizing the loss function for solving the PBE modeling the protein BphC is quite slow, a few hours for a large batch size. On the other hand, neural networks have strong representability and are capable in approximating different kinds of functions. Therefore, they are potentially useful for solving complex PDE problems, both high and low dimensional. Our extensive numerical tests have in fact shown that our neural network approach can handle relatively easily complex interface geometries. The weight transferability that we numerically found and possibly other transferrability properties of neural networks can be very useful in developing different machine learning methods for solving various kinds of PDEs problems such as those with parameters. We note that solving low-dimensional PDEs is a good testing process for developing machine learning methods for solving high-dimensional PDEs. With the development of advanced

training algorithms and optimized network architecture, we expect that the efficiency and accuracy of neural networks PDE solvers can be improved significantly.

(2) *Tuning hyper-parameters.* In general, there are many hyper-parameters in neural network simulations. In our neural network algorithm, such parameters include network architecture, the learning rate, and batch size. Currently, we determine all the hyper-parameters empirically, often following extensive numerical tests. We find that in general larger batch size leads to more accurate but more time consuming simulations and that the initial increase of the depth of network structure enhances the performance but further such increase will deteriorate the accuracy. More simulations and rigorous studies are needed to provide guidelines for choosing optimal values of these hyper-parameters. It is worth to mention that for different PDE problems, it is possible to use different network structures (e.g., not only with deep but also recurrent) and set a learning rate scheduler for adaptive training.

(3) *Training algorithms.* The development of efficient stochastic optimization algorithms is crucial to speed up the neural network training. Here, we have only tried the algorithm ADAM. It is desirable in future to test other algorithms. Moreover, convergence analysis and error estimates for neural network approximations of PDEs remain generally challenging. We can numerically verify the convergence of neural network simulations. But rigorous and non-conventional numerical analysis should be developed for neural network simulations in terms of the complexity of algorithm and rate of convergence with respect to neural network hyper-parameters, In this regard, our finding that the patch of near-loss-minimum on the loss surface may be large and flatten may help us develop nonconventional concepts and tools for the related convergence analysis.

(4) *Convergence.* The convergence of our neural network method is in the large limit of the penalty parameter λ, the number N_{sample} of sampling points (corresponding to our N_Ω and $N_{\partial\Omega}$) and the number $N_{iteration}$ of epochs (corresponding to our N_{iter}). We have proved the convergence of the electrostatic potential and energy of the penalized model as $\lambda \to \infty$. Our numerical tests show that the error of electrostatic potential decreases, while that of the energy does not clearly, as λ increases. The reason is that a larger penalty value λ leads to a higher value of the initial energy and hence requires more iteration steps to reduce further the energy. While computing accurately the electrostatic potential is our main objective, we understand that providing accurate estimates of the energy is also important. A possible improvement of accurately evaluating the energy will be to use the neural network simulated electrostatic potential and the numerical quadrature to calculate the integral that defines the energy. With a fixed value of the penalty parameter λ, our numerical tests consistently indicate the decrease of error in the electrostatic potential or the energy as the number of sampling points or the batch size increases. We point out that in practice one is prohibited to take a very large number of epochs as that can be very inefficiency. In particular, in the commonly used SGD algorithm, new sampling points are generated in new epochs. Thus, a very large number of epochs can significantly affect the efficiency.

(5) *CPU vs. GPU.* Since the primary objective of this work is to develop a new, machine learning approach to solving the PBE with application to biomolecular systems, we have only implemented our algorithms on CPUs. The GPU implementation may be necessary to speed up the neural network training as the loss function is often highly nonconvex and fluctuating. We will leave such implementation for future work.

(6) *Weight transferability.* The transferability of network weights that we have found in this study is a new feature for neural network simulations of complex systems. It may provide a novel and efficient route for large-scale molecular simulations, as network weight initialization can significantly affect the overall computational efficiency. Theoretically, it is an interesting and challenging question why a neural network algorithm has such a strong transferability, calling for further investigations.

Acknowledgment

This work was supported in part by the US National Science Foundation through the grant DMS-2208465 (BL), the National Natural Science Foundation of China through the grant 92470103, the Hong Kong RGC through the grant 17304324, and Seed Funding for Strategic Interdisciplinary Research Scheme 2021/22 (HKU) (ZZ).

References

[1] R. Adams. *Sobolev Spaces.* Academic Press, New York, 1975.

[2] R. Baron and J. A. McCammon. Molecular recognition and ligand association. *Annu. Rev. Phys. Chem.*, 64:151–175, 2013.

[3] J. Berg and K. Nyström. A unified deep artificial neural network approach to partial differential equations in complex geometries. *Neurocomput.*, 317:28–41, 2018.

[4] M. Born. Volumen und Hydratationswärme der Ionen. *Z. Phys.*, 1:45–48, 1920.

[5] O. Calin. *Deep Learning Architures.* Springer, 2020.

[6] J. Che, J. Dzubiella, B. Li, and J. A. McCammon. Electrostatic free energy and its variations in implicit solvent models. *J. Phys. Chem. B*, 112:3058–3069, 2008.

[7] H. B. Cheng, L.-T. Cheng, and B. Li. Yukawa-field approximation of electrostatic free energy and dielectric boundary force. *Nonlinearity*, 24(11):3215–3236, 2011.

[8] L.-T. Cheng, J. Dzubiella, J. A. McCammon, and B. Li. Application of the level-set method to the implicit solvation of nonpolar molecules. *J. Chem. Phys.*, 127:084503, 2007.

[9] I. L. Chern and Y. C. Shu. A coupling interface method for elliptic interface problems. *J. Comput. Phys.*, 225:2138–2174, 2007.

[10] M. E. Davis and J. A. McCammon. Electrostatics in biomolecular structure and dynamics. *Chem. Rev.*, 90:509–521, 1990.

[11] J. Dzubiella, J. M. J. Swanson, and J. A. McCammon. Coupling hydrophobicity, dispersion, and electrostatics in continuum solvent models. *Phys. Rev. Lett.*, 96:087802, 2006.

[12] J. Dzubiella, J. M. J. Swanson, and J. A. McCammon. Coupling nonpolar and polar solvation free energies in implicit solvent models. *J. Chem. Phys.*, 124:084905, 2006.

[13] W. E and B. Yu. The deep Ritz method: A deep learning-based numerical algorithm for solving variational problems. *Commun. Math. Stats.*, 6(1):1–12, 2018.

[14] L. C. Evans. *Partial Differential Equations*, volume 19 of *Graduate Studies in Mathematics.* Amer. Math. Soc., 2nd edition, 2010.

[15] E. Georgoulis, M. Loulakis, and A. Tsiourvas. Discrete gradient flow approximations of high dimensional evolution partial differential equations via deep neural networks. *Commun. Nonl. Sci. Numer. Simul.*, 117:106893, 2023.

[16] J. Han, A. Jentzen, and W. E. Solving high-dimensional partial differential equations using deep learning. *Proc. Natl Acad. Sci. USA*, 115(34):8505–8510, 2018.

[17] S. Haykin. *Neural Networks*. Pearson, 1999.

[18] W.-F. Hu, T.-S. Lin, and M.-C. Lai. A discontinuity capturing shallow neural network for elliptic interface problems. *J. Comput. Phys.*, 469:111576, 2022.

[19] Z. Huang and B. Li. Variational implicit solvation with Legendre-transformed Poisson–Boltzmann electrostatics. *Proc. R. Soc. A*, 480:20230731, 2024.

[20] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv:1412.6980*, 2014.

[21] I. Kwon, G. Jo, and K.-S. Shin. A deep network based on ResNet for predicting solutions of Poisson–Boltzmann equation. *MDPI Electronics*, 10:2627, 2021.

[22] C.-C. Lee. The charge conserving Poisson–Boltzmann equations: Existence, uniqueness, and maximum principle. *J. Math. Phys.*, 55:051503, 2014.

[23] B. Li. Minimization of electrostatic free energy and the Poisson–Boltzmann equation for molecular solvation with implicit solvent. *SIAM J. Math. Anal.*, 40:2536–2566, 2009.

[24] B. Li., X.-L. Cheng, and Z.-F. Zhang. Dielectric boundary force in molecular solvation with the Poisson–Boltzmann free energy: A shape derivative approach. *SIAM J. Applied Math*, 71(10):2093–2111, 2011.

[25] B. Li, Z. Zhang, and S. Zhou. The calculus of boundary variations and the dielectric boundary force in the Poisson–Boltzmann theory for molecular solvation. *J. Nonlinear Sci.*, 31(89):1–50, 2021.

[26] Y. Liao and P. Ming. Deep Nitsche method: Deep Ritz method with essential boundary conditions. *Commun. Comput. Phys.*, 29(5):1365–1384, 2021.

[27] Z. Liu, W. Cai, and Z.-Q. J. Xu. Multi-scale deep neural network (MscaleDNN) for solving Poisson–Boltzmann equation in complex domains. *Commun. Comput. Phys.*, 28(5):1970–2001, 2020.

[28] B. Lu, X. Cheng, J. Huang, and J. A. McCammon. Order N algorithm for computation of electrostatic interaction in biomolecular systems. *Proc. Natl. Acad. Sci. USA*, 103:19314–19319, 2006.

[29] B. Z. Lu, Y. C. Zhou, M. J. Holst, and J. A. McCammon. Recent progress in numerical methods for the Poisson-Boltzmann equation in biophysical applications. *Commun. Comput. Phys.*, 3:973–1009, 2008.

[30] S. Mahan, E. J. King, and A. Cloninger. Nonclosedness of sets of neural networks in Sobolev spaces. *Neural Networks*, 137:85–96, 2021.

[31] Y. Marcus. Thermodynamics of solvation of ions. Part 5.– Gibbs free energy of hydration at 298.15 K. *J. Chem. Soc. Faraday Trans.*, 87:2995–2999, 1991.

[32] M. Mirzadeh, M. Theillard, A. Helgadöttir, D. Boy, and F. Gibou. An adaptive, finite difference solver for the nonlinear Poisson–Boltzmann equation with applications to biomolecular computations. *Commun. Comput. Phys.*, 13(1):150–173, 2013.

[33] P. Petersen, M. Raslan, and F. Voigtlaender. Topological properties of the set of functions generated by neural networks of fixed size. *Found. Comp. Math.*, 21:375–444, 2021.

[34] Y. Ren, S. Amihere, W. Geng, and S. Zhao. Comparison of three matched interface and boundary (MIB) schemes for solving the nonlinear Poisson–Boltzmann equation. *Commun. Info. Systems*, 24:231–251, 2024.

[35] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *Nature*, 323:533–536, 1986.

[36] T. Senda, K. Sugiyama, H. Narita, T. Yamamoto, K. Kimbara, M. Fukuda, M. Sato, K. Yano, and Y. Mitsui. Three-dimensional structures of free form and two substrate complexes of an extradiol ring-cleavage type Dioxygenase, the BphC enzyme from Pseudomonassp. strain KKS102. *J. Mol. Biol.*, 255:735–752, 1996.

[37] K. A. Sharp and B. Honig. Electrostatic interactions in macromolecules: Theory and applications. *Annu. Rev. Biophys. Biophys. Chem.*, 19:301–332, 1990.

[38] J. Sirignano and K. Spiliopoulos. DGM: A deep learning algorithm for solving partial differential equations. *J. Comput. Phys.*, 375:1339–1364, 2018.

[39] S. L. Sobolev. *Applications of Functional Analysis in Mathematical Physics.* Amer. Math. Soc., 1963. Translated from the Russian by F. E. Browder.

[40] D. N. Tanyu, J. Ning, T. Freudenberg, N. Heilenkötter, A. Rademacher, and U. I. P. Maass. Deep learning methods for partial differential equations and related parameter identification problems. *Inverse Problems*, 39:103001, 2023.

[41] L. Wang, B. J. Berne, and R. A. Friesner. Ligand binding to protein-binding pockets with wet and dry regions. *Proc. Natl. Acad. Sci. USA*, 108:1326–1330, 2011.

[42] Z. Wang, J. Che, L.-T. Cheng, J. Dzubiella, B. Li, and J. A. McCammon. Level-set variational implicit solvation with the Coulomb-field approximation. *J. Chem. Theory Comput.*, 8:386–397, 2012.

[43] Z. Wang and Z. Zhang. A mesh-free method for interface problems using the deep learning approach. *J. Comput. Phys.*, 400:108963, 2020.

[44] H. Wei, R. Luo, and R. Qi. An efficient second-order Poisson–Boltzmann method. *J. Comput. Chem.*, 40(12):1257–1269, 2019.

[45] L. Wilson, W. Geng, and R. Krasny. TABI-PB 2.0: An improved version of the Treecode-Accelerated Boundary Integral Poisson–Boltzmann solver. *J. Phys. Chem. B*, 126:7104–7113, 2022.

[46] R. Zhou, X. Huang, C. J. Margulis, and B. J. Berne. Hydrophobic collapse in multidomain protein folding. *Science*, 305:1605–1609, 2004.

[47] S. Zhou, L.-T. Cheng, J. Dzubiella, B. Li, and J. A. McCammon. Variational implicit solvation with Poisson–Boltzmann theory. *J. Chem. Theory Comput.*, 10:1454–1467, 2014.

[48] S. Zhou, R. G. Weiß, L.-T. Cheng, J. Dzubiella, J. A. McCammon, and B. Li. Variational implicit-solvent predictions of the dry–wet transition pathways for ligand–receptor binding and unbinding kinetics. *Proc. Natl. Acad. Sci. U.S.A.*, 116(30):14989–14994, 2019.