# Machine learning for performance enhancement of molecular dynamics simulations *

JCS Kadupitiya[1], Geoffrey C. Fox[1], and Vikram Jadhao[1][0000−0002−8034−2654]

Intelligent Systems Engineering, Indiana University, Bloomington IN 47408, USA
{kadu,gcf,vjadhao}@iu.edu

**Abstract.** We explore the idea of integrating machine learning with simulations to enhance the performance of the simulation and improve its usability for research and education. The idea is illustrated using hybrid OpenMP/MPI parallelized molecular dynamics simulations designed to extract the distribution of ions in nanoconfinement. We find that an artificial neural network based regression model successfully learns the desired features associated with the output ionic density profiles and rapidly generates predictions that are in excellent agreement with the results from explicit molecular dynamics simulations. The results demonstrate that the performance gains of parallel computing can be further enhanced by using machine learning.

**Keywords:** Machine Learning · Molecular Dynamics Simulations · Parallel Computing · Scientific Computing · Clouds

## 1 Introduction

In the fields of physics, chemistry, bioengineering, and materials science, it is hard to overstate the importance of parallel computing techniques in providing the needed performance enhancement to carry out long-time simulations of systems of many particles with complex interaction energies. These enhanced simulations have enabled the understanding of microscopic mechanisms underlying the macroscopic material and biological phenomena. For example, in a typical molecular dynamics (MD) simulation of ions in nanoconfinement [2, 13] (Figure 1), $\approx 1$ nanosecond of dynamics of $\approx 500$ ions on one processor takes $\approx 12$ hours of runtime, which is prohibitively large to extract converged results for ion distributions within reasonable time frame. Performing the same simulation on a single node with multiple cores using OpenMP shared memory reduces the runtime by a factor of 10 ($\approx 1$ hour), enabling simulations of the same system for longer physical times. Using MPI dramatically enhances the simulation performance: for systems with thousands of ions, speedup of over 100 can be achieved, enabling the generation of the needed data for evaluating converged ionic distributions. Further, a hybrid OpenMP/MPI approach can provide even higher speedup of over 400 for similar-size systems enabling state-of-the-art research

* Supported by National Science Foundation through Awards 1720625 and 1443054.

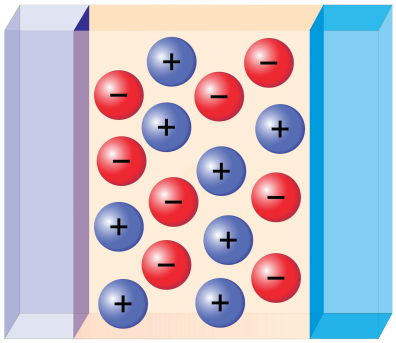

**Fig. 1.** Sketch of ions represented by blue and red spheres confined by material surfaces.

with simulations that can explore the dynamics of ions with less controlled approximations, accurate potentials, and for tens or hundreds of nanoseconds over a wider range of physical parameters.

Despite the employment of the optimal parallelization model suited for the size and complexity of the system, scientific simulations remain time consuming. This is particularly evident in the area of using simulations in education where real-time simulation-driven responses to students in classroom settings are desirable. While in research settings, instantaneous results are generally not expected and simulations can take up to several days, it is often desirable to rapidly access expected trends associated with relevant physical quantities that could have been learned from past simulations or could be predicted with reasonable accuracy based on the history of data generated from earlier simulation runs. As an example, consider a simulation framework, Ions in Nanoconfinement, that we deployed as a web application on nanoHUB to execute the aforementioned simulations of ionic systems [14]. In classroom usage, we have observed that the fastest simulations can take about 10 minutes to provide the converged ionic densities while the slowest ones (typically associated with larger system sizes and more complex ion-ion interaction energies) can take over 3 hours. Similarly, for research applications, not having rapid access to expected overall trends in the key simulation output quantities (e.g., the variation of contact density as a function of ion concentration) can make the process of starting new investigations unwieldy and time-consuming. Primary factors that contribute to this scenario are the time delays resulting from the combination of waiting time in a queue on a computing cluster and the actual runtime for the simulation.

In this paper, we explore the idea of integrating machine learning (ML) layer with simulations to enhance the performance and improve the usability of simulations for both research and education. This idea is inspired by the recent development and use of ML in material simulations and scientific software applications [4, 17, 6, 24, 9]. We employ a particular example, hybrid OpenMP/MPI parallelized MD simulations of ions in nanoconfinement [13, 14] to illustrate this idea. We demonstrate that an artificial neural network (ANN) based regres-

sion model, trained on data generated via these simulations, successfully learns pre-identified key features associated with the output ionic density profile (the contact, mid-point, and peak densities). The ML approach entirely bypasses simulations and generates predictions that are in excellent agreement with results obtained from explicit MD simulations. The results demonstrate that the performance gains of parallel computing can be enhanced using data-driven approaches such as ML which improves the usability of the simulation framework by enabling real-time engagement and anytime access.

## 2 Background and Related Work

### 2.1 Coarse-grained simulations of ions in nanoconfinement

The distribution of ions often determines the assembly behavior of charged or neutral nanomaterials such as nanoparticles, colloids, or biological macromolecules. Accurate knowledge of this ionic structure is exploited in many applications [8] including the design of double-layer supercapacitor and the extraction of metal ions from wastewater. As a result, extracting the distribution of ions in nanoconfinement created by material surfaces has been the focus of recent experiments and computational studies [18, 22, 25, 21, 13]. From a modeling standpoint, the surfaces are often treated as planar interfaces considering the size difference between the ions and the confining material particles, and the solvent is coarse-grained to speed-up the simulations. Such coarse-grained simulations have been employed to extract the ionic distributions over a wide range of electrolyte concentrations, ion valencies, and interfacial separations using codes developed in individual research groups [2, 3, 13] or using general purpose software packages such as ESPRESSO [16] and LAMMPS [19].

Generally, the average equilibrium ionic distribution, resulting from the competing inter-ionic steric and electrostatic interactions as well as the interactions between the ions and surfaces, is a quantity of interest. However, in many cases, the density of ions at the point of closest approach of the ion to the interface (contact density), the peak density, or the density at the center of the slit (mid-point density) directly relate to important experimentally-measurable quantities such as the effective force between the confining surfaces or the osmotic pressure [25, 21]. It is thus useful to compute the variation of the contact, peak, and mid-point densities as a function of the solution conditions or ionic attributes.

### 2.2 Machine learning for enhancing simulation performance

Recent years have seen a surge in the use of ML to accelerate computational techniques aimed at understanding material phenomena. ML has been used to predict parameters, generate configurations in material simulations, and classify material properties [4, 9, 17, 24]. For example, Fu et al. [17] employed ANN to select efficient updates to accelerate Monte Carlo simulations of classical Ising spin models near critical parameters associated with the phase transition. Similarly, Botu et al. [4] employed kernel ridge regression to accelerate MD method

for nuclei-electron systems by learning the selection of probable configurations in MD simulations, which enabled bypassing explicit simulations for several steps.

The integration of ML layer for performance enhancement of scientific simulation frameworks deployed as web applications is relatively far less explored. nanoHUB is the largest online resource for education and research in nanotechnology [15]. This cyberinfrastructure hosts over 500 simulation tools and serves 1.4 million users worldwide. Our survey indicated that only one simulation tool on nanoHUB [10] employs ML-based methods to enhance the performance and usability of the simulation software. This simulation tool employs a deep neural network to bypass computational limitations in extracting transfer times associated with the excitation energy transport in light-harvesting systems [10].

## 3 Framework for Simulating Ions in Nanoconfinement

We developed a framework for simulating ions in nanoscale confinement (referred here as the nanoconfinement framework) that enabled a systematic investigation of the self-assembly of ions characterized by different ionic attributes (e.g., ion valency) and solution conditions (e.g., ion concentration and inter-surface separation) [11, 12, 23, 13, 14]. The framework has been employed to extract the ionic structure in electrolyte solutions confined by planar and spherical surfaces. Results have elucidated the microscopic mechanisms involving the ionic correlations and steric effects that determine the distribution of ions [11–13].

In the work presented here, we focus on ions confined by unpolarizable surfaces where the simulations are relatively faster [13], thus easing the training and testing of the ML model. We identify the following system attributes as key parameters that determine the self-assembly of ions: inter-surface separation or confinement length $h$, ion valencies ($z_p$, $z_n$ associated with the positive and negative ions), electrolyte concentration $c$, and ion diameter $d$. We ignore the effects arising due to the solvent-induced asymmetry in ionic sizes, and assign the positive and negative ions with the same diameter. Additional details regarding the model and simulation method can be found in the original paper [13].

The nanoconfinement framework employs a code written in C++ for simulating ions near unpolarizable interfaces. The code, available as an open-source repository on GitHub [14], is accelerated using a hybrid parallel programming technique (see Sec. 3.1). This framework is also deployed as a web application (Ions in Nanoconfinement) on nanoHUB [14] (see Sec. 3.2). We have verified that the ionic density profiles obtained using our code agree with the densities extracted via simulations performed in LAMMPS. We note that the ML model proposed in this work to predict the desired key ionic densities is agnostic to the code engine that enables the MD simulations for generating the training dataset.

### 3.1 Hybrid OpenMP/MPI parallelization

The nanoconfinement framework is based on a code that employs the velocity-Verlet algorithm to update the positions and velocities at each simulation step.

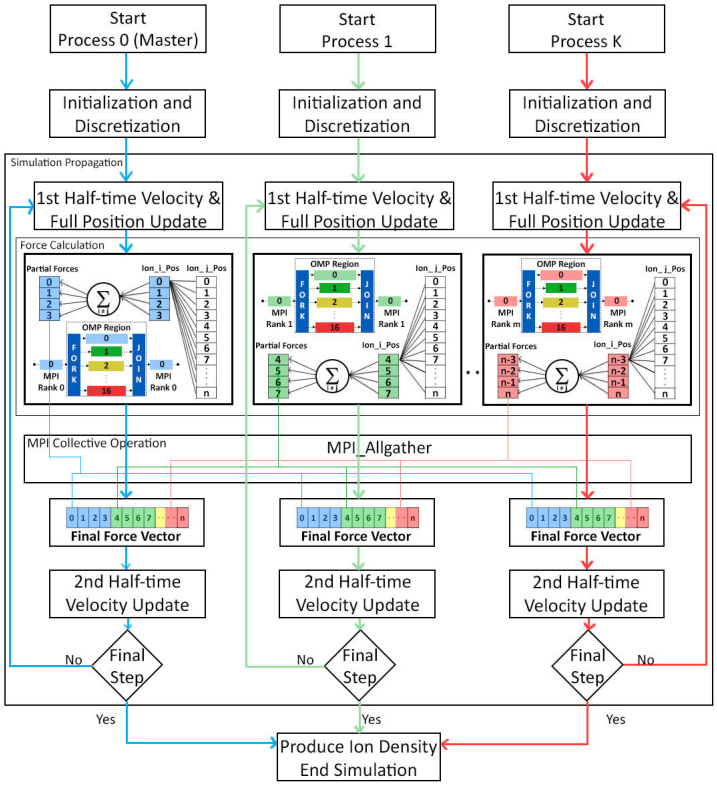


**Fig. 2.** Hybrid model implemented in the Force Calculation block using MPI and OpenMP to accelerate the nanoconfinement framework.

First, the velocities of all ions are updated for half timestep $\Delta/2$, following which the positions of all ions are updated for $\Delta$. At this point, the forces on all ions are computed, and finally the velocities for all ions are updated for the next $\Delta/2$. Once the system reaches equilibrium, the positions of the ions are stored periodically to extract density profiles.

To improve the runtime of the simulation, the framework employs the hybrid master-only model that uses one MPI process per node and OpenMP on the cores of the node, with no MPI calls inside the parallel regions. This hybrid model is applied for the force and energy calculation subroutines, and it enables the domain decomposition under a two-level mechanism. On the MPI level, coarse-grained domain decomposition is performed using boundary conditions as explained in Figure 2. The second level of domain decomposition is achieved through OpenMP loop level parallelization inside each MPI process. This multilevel domain decomposition has advantages over pure MPI or pure OpenMP, when cache performance is taken into consideration. This strategy

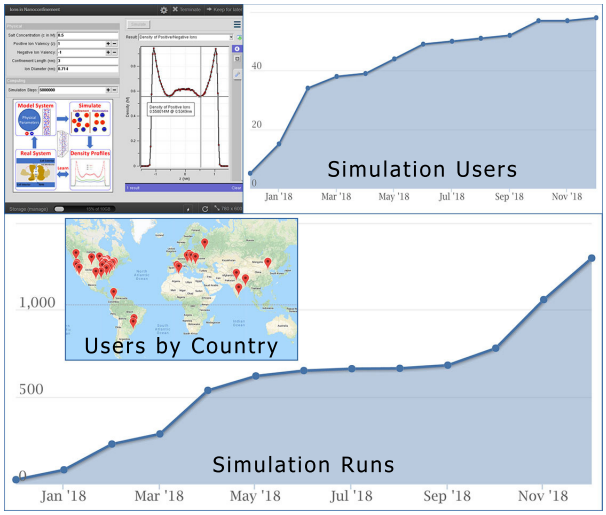


**Fig. 3.** A screenshot of the GUI (top left) and usage statistics of the nanoconfinement framework deployed on nanoHUB.

also provides the maximum access locality, a minimum number of cache misses, non-uniform memory access (NUMA) traffic and inter-node communication [20].

### 3.2 Deployment on nanoHUB

The nanoconfinement framework is deployed as a web application (Ions in Nanoconfinement) [14] on the nanoHUB cyberinfrastructure. nanoHUB provides user-friendly, web-based access for executing simulation codes to researchers, students, and educators broadly interested in nanoscale engineering [15]. In less than 1 year of its launch, the Ions in Nanoconfinement application has nucleated 58 users worldwide and has been run over 1200 times [14]. This application is designed to launch simulations that use virtual machines or supercomputing clusters depending on user-selected inputs, and it has been employed to teach graduate courses at Indiana University. Advances in the framework that enable real-time results can significantly enhance the experience of users that employ this application in both education and research. The current version of "Ions in Nanoconfinement", like almost all other applications on nanoHUB, does not support this feature.

## 4 ML-enabled Performance Enhancement

We now describe an approach based on ML to enhance the overall performance and usability of the nanoconfinement framework. Figure 4 shows the overview of the implemented methodology. First, the attributes of the ions and solution conditions that characterize the system are fed to the nanoconfinement

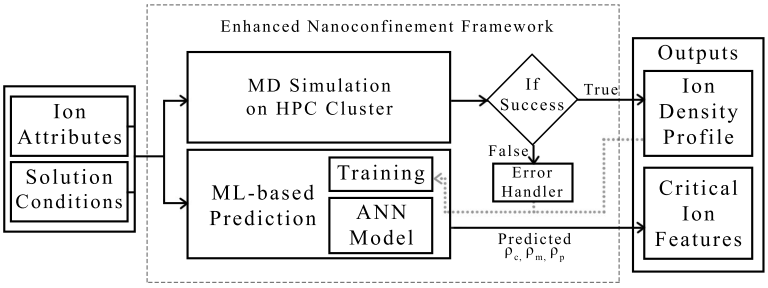


**Fig. 4.** System overview of the enhanced nanoconfinement framework.

framework. These inputs are used to launch the MD simulation on the high-performance computing (HPC) cluster. Simultaneously, these inputs are also fed to the ML-based density prediction module to predict the contact, mid-point, and peak densities. The outputs of the MD simulation are the ion density profiles that characterize the ionic structure near unpolarizable interfaces. Error handler aborts the program and displays appropriate error messages when a simulation fails due to any pre-defined criteria. In addition, at the end of the simulation run, ionic density values are saved for future retraining of the ML model. ML model is retrained for every 2500 new simulation runs.

After reviewing and experimenting with many ML techniques for parameter tuning and prediction including polynomial regression, support vector regression, decision tree regression, and random forest regression, the artificial neural network (ANN) was adopted for predicting critical features associated with the output ionic density. Figure 5 shows the details of this ANN-based ML model. The data preparation and preprocessing techniques, feature extraction and regression techniques as well as their validation are discussed below.

### 4.1 Data preparation and preprocessing

Prior domain experience and backward elimination using the adjusted R squared is used for creating the training data set. Five input parameters that significantly affect the dynamics of the system are identified: confinement length $h$, positive valency $z_p$, negative valency $z_n$, salt concentration $c$, and the diameter of the ions $d$. Future work will explore the training with additional input parameters such as temperature and solvent permittivity that are fixed in this initial study to room temperature (298 K) and water permittivity ($\approx 80$) respectively.

Contact density $\rho_c$, mid-point (center of the slit) density $\rho_m$, and peak density $\rho_p$ associated with the final (converged) distribution for positive ions were selected as the output parameters. Few discrete values for each of the input/output parameters were experimented with and swept over to create and run 6,864 simulations for training the ML model. The range for ionic system parameters was selected based on physically meaningful and experimentally-relevant values: $h \in (3.0, 4.0)$ nm, $z_p \in 1, 2, 3$ (in units of electronic charge $|e|$); $z_n \in -1, -2$

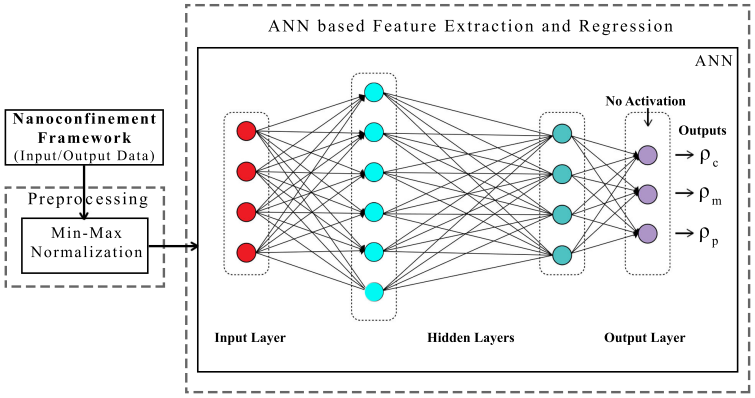


**Fig. 5.** ANN-based regression model to enhance the nanoconfinement framework.

(in units of $|e|$); $c \in (0.3, 0.9)$ M, and $d \in (0.5, 0.75)$ nm. All simulations were performed for over $\approx 5$ nanoseconds. The entire data set was separated into training and testing sets using a ratio of 0.7:0.3. Min−max normalization filter was applied to normalize the input data at the preprocessing stage.

### 4.2 Feature extraction and regression

The ANN algorithm with two hidden layers (Figure 5) was implemented in Python for regression of three continuous variables in the ML model. Outputs of the hidden layers were wrapped with the *relu* function; the latter was found to converge faster compared to the sigmoid function. No wrapping functions were used in the output layers of the algorithm as ANN was trained for regression.

By performing a grid search, hyper-parameters such as the number of first hidden layer units, second hidden layer units, batch size, and the number of epochs were optimized to 17, 9, 25, and 100 respectively. Adam optimizer was used as the backpropagation algorithm. The weights in the hidden layers and in the output layer were initialized to random values using a normal distribution at the beginning. The mean square loss function was used for error calculation. To stop overtraining the network, a drop out mechanism for hidden layer neurons was employed during the training time. ANN implementation, training, and testing were programmed using scikit-learn, Keras, and TensorFlow ML libraries [7, 5, 1]. Rest of the regression methods were implemented using scikit-learn ML libraries. For the kernel ridge regression, radial basis function kernel was used.

## 5 Results

### 5.1 Bypassing simulations with ML-enabled predictions

We experimented with 6 regression models to predict the key output density features identified above: contact density ($\rho_c$), mid-point density ($\rho_m$), and

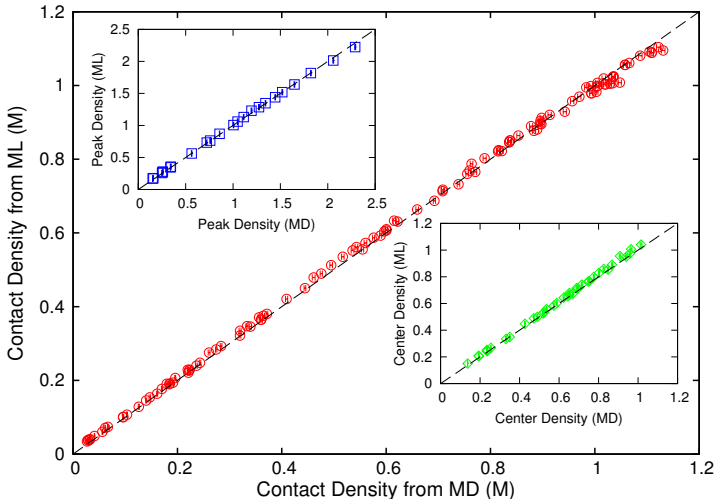


**Fig. 6.** Accuracy comparison between ML predictions and MD simulation results for the contact densities (red circles) of ions in systems characterized by inputs selected from the following ranges of parameters: $h \in (3.0, 4.0)$ nm, $z_p \in 1, 2, 3$, $z_n \in -1, -2$, $c \in (0.3, 0.9)$ M, and $d \in (0.5, 0.75)$ nm. Top-left and bottom-right insets show the comparison for the peak (blue squares) and mid-point (green diamonds) densities respectively for a subset of the selected systems. Black dashed lines with a slope of 1 represent perfect correlation. All densities are shown in units of molars.

peak density ($\rho_p$). These models were tested on 2060 sets of input parameters $(h, z_p, z_n, c, d)$. These sets were comprised of parameter values within the range for which the models were trained; see Section 4.1. Table 1 shows the success rate and the mean square error (MSE) for testing data sets. The success rate was calculated based on the error bars associated with the density values obtained via MD simulations: ML prediction was considered successful when the predicted density value was within the error bar of the simulation estimate. Simulations were run for sufficiently long times (over $\approx 5$ nanoseconds) to obtain converged density estimates and error bars. MSE values are calculated using k-fold cross-validation techniques with k = 20. ANN based regression model predicted $\rho_c$, $\rho_m$ and $\rho_p$ accurately with a success rate of 95.52% (MSE $\approx 0.0000718$), 92.07% (MSE $\approx 0.0002293$), and 94.78% (MSE $\approx 0.0002306$) respectively. ANN outperformed all other non-linear regression models (Table 1).

Figure 6 shows the comparison between the predictions made by the ML model and the results obtained from MD simulations for the contact, mid-point, and peak densities associated with positive ions. For clarity, results are shown for a randomly selected subset of the entire testing dataset described in Section 4.1. $\rho_c$, $\rho_m$ and $\rho_p$ predicted by the ML model were found to be in excellent agreement with those calculated using the MD method; data from either approach fall on the dashed lines which indicate perfect correlation.

**Table 1.** Comparison of regression models for the prediction of output density values.

| Model | Contact Density | | Midpoint Density | | Peak Density | |
|---|---|---|---|---|---|---|
| | *Success %* | *MSE* | *Success %* | *MSE* | *Success %* | *MSE* |
| Polynomial | 61.04 | 0.0129300 | 60.84 | 0.0187700 | 61.87 | 0.0100400 |
| Kernel-Ridge | 78.86 | 0.0030900 | 76.57 | 0.0041200 | 75.93 | 0.0049800 |
| Support Vector | 80.11 | 0.0012700 | 79.55 | 0.0024900 | 81.98 | 0.0010600 |
| Decision Tree | 68.44 | 0.0084600 | 64.54 | 0.0094900 | 62.47 | 0.0110700 |
| Random Forest | 74.15 | 0.0045700 | 70.85 | 0.0078900 | 75.09 | 0.0040800 |
| ANN based | 95.52 | 0.0000718 | 92.07 | 0.0002293 | 94.78 | 0.0002306 |

### 5.2 Rapid access to trendlines using ML

Trendlines exhibiting the variation of $\rho_c$, $\rho_m$, and $\rho_p$ for a wide range and combinations of the five input parameters ($h$, $z_p$, $z_n$, $c$, $d$) were extracted using the ML model. In Figures 7 and 8, we show a small selected subset of these trends.

Figure 7 shows the variation of the contact density $\rho_c$ with the ion diameter $d$ and confinement length $h$. Figure 7 (top) illustrates how $\rho_c$ varies with $d \in (0.5, 0.75)$ nm at constant $h = 4$ nm, $z_p = 3$, $z_n = -1$, and $c = 0.85$ M. Figure 7 (bottom) illustrates the variation in $\rho_c$ when the confinement length $h$ is tuned between 3.0 and 4.0 nm, with other parameters held constant ($z_p = 2$, $z_n = -1$,

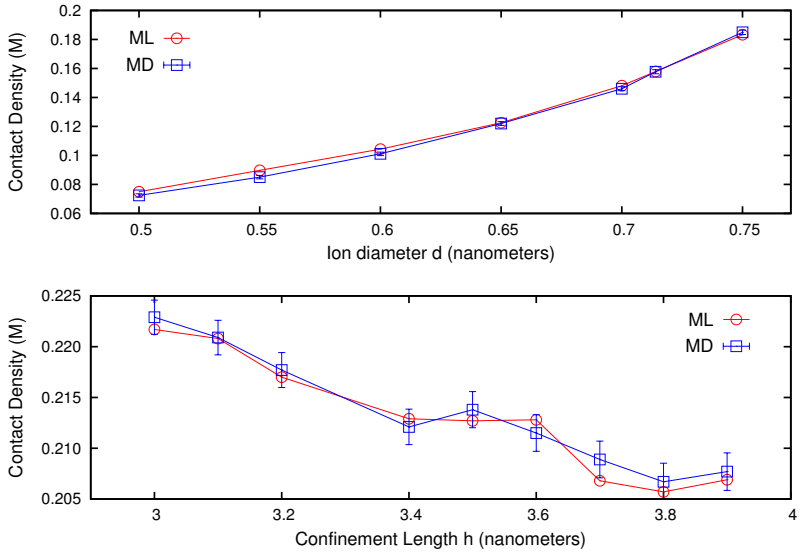


**Fig. 7.** (Top) Trendlines for contact density vs. ion diameter for systems with $h = 4$ nm, $z_p = 3, z_n = -1$, and $c = 0.85$ M. (Bottom) Trendlines for contact density vs. confinement length for systems with $z_p = 2$, $z_n = -1$, $c = 0.9$ M, and $d = 0.553$ nm.

$c = 0.9$ M, and $d = 0.553$ nm). Circles represent ML predictions and squares show MD results. ML predictions are within the errorbars generated via MD simulations and follow the simulation-predicted trends for both cases. Results demonstrate that contact density varies rapidly when the diameter is changed but exhibits a slower variation when the confinement length is varied. We find that the same ML model is able to track the distinct variations while exhibiting different resolution (sensitivity) criteria.

Figure 8 shows similar comparison between ML and MD results for the variation of $\rho_c$, $\rho_m$ and $\rho_p$ vs. salt concentration $c$. Excellent agreement is seen between the two approaches. We note that the prediction time for the ML model to obtain these densities is in the order of a few seconds while the simulations can take up to hours to compute one contact density with similar accuracy level.

### 5.3 Speedup

Traditional speedup formulae associated with parallel computing methods need to be adapted for evaluating the speedup associated with the ML-enhanced simulations. We propose the following simple formula which is illustrative at this preliminary stage. We define the ML speedup as:

$$S = \frac{t_{sim}}{t_p + t_{tr} \cdot N_{tr}/N_p}, \tag{1}$$

where $t_{sim}$ is the time to run the MD simulation via the sequential model, $t_p$ is the time for the ML model to perform a forward propagation for one set of

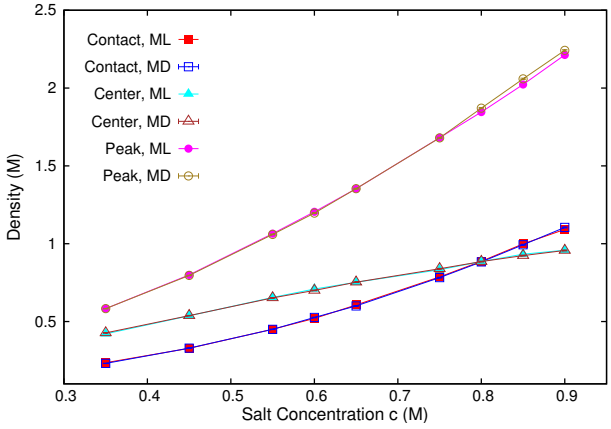


**Fig. 8.** Contact, peak, and center-of-the-slit (mid-point) density vs. salt concentration associated with the distribution of ions for systems characterized with $h = 3.2$ nm, $z_p = 1$, $z_n = -1$, and $d = 0.714$ nm. Closed symbols represent ML predictions and open symbols with error bars are MD simulation results.

inputs (prediction or "lookup" time), $N_p$ is the number of predictions made using the ML model, $N_{tr}$ is the number of elements in the training dataset, and $t_{tr}$ is the average MD simulation walltime to create one of these elements. $N_{tr}t_{tr}$ represents the total time to create the training dataset and it is much larger than the TensorFlow training time.

The above formula highlights the key feature of the ML-based approach: the speedup $S$ increases as ML model is used to make more predictions, that is, $S$ rises with increasing $N_p$. As $N_p$ approaches infinity, $S$ approaches $t_{sim}/t_p$; for our MD simulations ($t_{sim} \approx 12$ hours) and ANN model ($t_p \approx 0.25$ seconds), we find this ratio to be over $10^5$. On the other hand, if the number of predictions made through the ML model is small (smaller $N_p$), then $S$ is expected to be small. We explore this scenario further. For implementing our ML model, the training dataset consisted of 4804 simulation configurations, making $N_{tr} = 4804$. The time $t_{tr}$ to generate one element of this training set is similar to the average runtime of the parallelized MD simulation; we find $t_{tr} \approx t_{sim}/100$. Using these relations and noting $t_p \ll t_{sim}$, a lower bound on speedup can be derived as the result for $N_p = 1$. For this case, we find the "speedup" $S \approx t_{sim}/(t_{tr}N_{tr}) \approx 10^{-2}$. Finally, when the number of predictions $N_p$ are similar to the number of elements in the training dataset $N_{tr}$, then Eq. 1 yields $S \approx t_{sim}/t_{tr}$, which is equivalent to the speedup associated with the traditional parallel computing approach.

## 6 Outlook and Future Work

Based on the aforementioned investigations, we propose to design and integrate an ML layer with the nanoconfinement framework. This enhanced application will be deployed on nanoHUB using the Jupyter python notebook interface. Figure 9 shows a sketch of the proposed GUI. Users will be able to click both "Run with MD" button and "Predict with ML" button simultaneously or separately depending on the desired information. "Predict with ML" will activate the ML layer and predict $\rho_c$, $\rho_m$, and $\rho_p$ almost instantaneously. These ML-predicted values will be shown in three text boxes and will appear as markers on the density profile plot. If users also select "Run with MD", the entire density profile will be added at the end of the simulation. For illustration purposes, Figure 9 shows the final density plot using this integrated MD + ML approach for the input parameters $h = 3.0$ nm, $z_p = 1$, $z_n = -1$, $c = 0.9$ M, and $d = 0.714$ nm.

In this initial study, we focused on a particular example framework to illustrate the idea of using ML-based methods to enhance the performance and usability of scientific simulations. The results from this investigation are encouraging and we intend to explore these ideas in the future to provide a richer set of predictions (finer-grained density profile) for the nanoconfinement framework, and extend the method to other simulation techniques. Here, the training data was created as a distinct step during the process of generating the ML model. Future approaches will involve implicit training of the ANN while the simulations are used in research and education. We expect that the usefulness of the ML-enabled enhancements demonstrated in this work strengthen the case for sci-

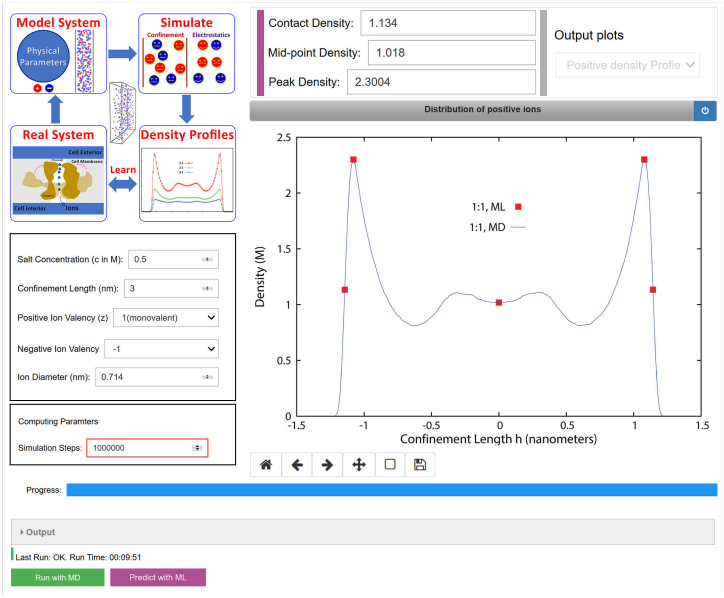


**Fig. 9.** Proposed GUI for integrating an ML layer with the nanoconfinement framework deployed on nanoHUB. The GUI includes text boxes (top) showing the ML-predicted contact, mid-point, and peak densities for an example ionic system. These results also appear as markers on the plot showing the density profile generated by MD simulations.

entific simulation applications to be designed and developed with an ML wrapper that both optimizes the application execution and learns from the simulations.

## References

1. Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., et al.: Tensorflow: a system for large-scale machine learning. In: OSDI. vol. 16, pp. 265–283 (2016)
2. Allen, R., Hansen, J.P., Melchionna, S.: Electrostatic potential inside ionic solutions confined by dielectrics: a variational approach. Phys. Chem. Chem. Phys. **3**, 4177–4186 (2001)
3. Boda, D., Gillespie, D., Nonner, W., Henderson, D., Eisenberg, B.: Computing induced charges in inhomogeneous dielectric media: Application in a monte carlo simulation of complex ionic systems. Phys. Rev. E **69**(4), 046702 (Apr 2004)
4. Botu, V., Ramprasad, R.: Adaptive machine learning framework to accelerate ab initio molecular dynamics. International Journal of Quantum Chemistry **115**(16), 1074–1083 (2015)
5. Buitinck, L., et al.: Api design for machine learning software: experiences from the scikit-learn project. arXiv:1309.0238 (2013)
6. Butler, K.T., Davies, D.W., Cartwright, H., Isayev, O., Walsh, A.: Machine learning for molecular and materials science. Nature **559**(7715), 547 (2018)
7. Chollet, F., et al.: Keras (2015)

8. Feng, G., Qiao, R., Huang, J., Sumpter, B.G., Meunier, V.: Ion distribution in electrified micropores and its role in the anomalous enhancement of capacitance. ACS Nano **4**(4), 2382–2390 (2010)
9. Ferguson, A.L.: Machine learning and data science in soft materials engineering. Journal of Physics: Condensed Matter **30**(4), 043002 (2017)
10. Häse, F., Kreisbeck, C., Aspuru-Guzik, A.: Machine learning for quantum dynamics: deep learning of excitation energy transfer properties. Chemical science **8**(12), 8419–8426 (2017)
11. Jadhao, V., Solis, F.J., Olvera de la Cruz, M.: Simulation of charged systems in heterogeneous dielectric media via a true energy functional. Phys. Rev. Lett. **109**, 223905 (Nov 2012)
12. Jadhao, V., Solis, F.J., Olvera de la Cruz, M.: A variational formulation of electrostatics in a medium with spatially varying dielectric permittivity. The Journal of Chemical Physics **138**(5), 054119 (2013)
13. Jing, Y., Jadhao, V., Zwanikken, J.W., Olvera de la Cruz, M.: Ionic structure in liquids confined by dielectric interfaces. The Journal of chemical physics **143**(19), 194508 (2015)
14. Kadupitiya, K., Marru, S., Fox, G.C., Jadhao, V.: Ions in nanoconfinement (Dec 2017), https://nanohub.org/resources/nanoconfinement, online on nanoHUB; source code on GitHub at github.com/softmaterialslab/nanoconfinement-md
15. Klimeck, G., McLennan, M., Brophy, S.P., III, G.B.A., Lundstrom, M.S.: nanohub.org: Advancing education and research in nanotechnology. Computing in Science Engineering **10**(5), 17–23 (Sept 2008)
16. Limbach, H.J., Arnold, A., Mann, B.A., Holm, C.: ESPResSo – an extensible simulation package for research on soft matter systems. Comp. Phys. Comm. **174**(9), 704–727 (May 2006)
17. Liu, J., Qi, Y., Meng, Z.Y., Fu, L.: Self-learning monte carlo method. Phys. Rev. B **95**, 041101 (Jan 2017)
18. Luo, G., Malkova, S., Yoon, J., Schultz, D.G., Lin, B., Meron, M., Benjamin, I., Vansek, P., Schlossman, M.L.: Ion distributions near a liquid-liquid interface. Science **311**(5758), 216–218 (2006)
19. Plimpton, S.: Fast parallel algorithms for short-range molecular dynamics. Journal of Computational Physics **117**(1), 1 – 19 (1995)
20. Rabenseifner, R., Hager, G., Jost, G.: Hybrid mpi/openmp parallel programming on clusters of multi-core smp nodes. In: 2009 17th Euromicro International Conference on Parallel, Distributed and Network-based Processing. pp. 427–436. IEEE (2009)
21. dos Santos, A.P., Netz, R.R.: Dielectric boundary effects on the interaction between planar charged surfaces with counterions only. The Journal of chemical physics **148**(16), 164103 (2018)
22. Smith, A.M., Lee, A.A., Perkin, S.: The electrostatic screening length in concentrated electrolytes increases with concentration. The journal of physical chemistry letters **7**(12), 2157–2163 (2016)
23. Solis, F.J., Jadhao, V., Olvera De La Cruz, M.: Generating true minima in constrained variational formulations via modified lagrange multipliers. Physical Review E **88**(5), 053306 (2013)
24. Spellings, M., Glotzer, S.C.: Machine learning for crystal identification and discovery. AIChE Journal **64**(6), 2198–2206 (2018)
25. Zwanikken, J.W., Olvera de la Cruz, M.: Tunable soft structure in charged fluids confined by dielectric interfaces. Proceedings of the National Academy of Sciences **110**(14), 5301–5308 (2013)