Dynamically polarisable force-fields for surface simulations via multi-output classification Neural Networks

Nicodemo Di Pasquale,[*,†] Joshua D. Elliott,[†] Panagiotis Hadjidoukas,[‡] and Paola Carbone[†]

[†]*Department of Chemical Engineering and Analytical Science, University of Manchester, Manchester M13 9AL, United Kingdom*

[‡]*IBM Research, Zürich, Switzerland*

E-mail: nicodemo.dipasquale@manchester.ac.uk

Abstract

We present a general procedure to introduce electronic polarization into classical Molecular Dynamics (MD) force-fields using a Neural Network (NN) model. We apply this framework to the simulation of a solid-liquid interface where the polarization of the surface is essential to correctly capture the main features of the system. By introducing a multi-input, multi-output NN and treating the surface polarization as a discrete classification problem we are able to obtain very good accuracy in terms of quality of predictions. Through the definition of a custom loss function we are able to impose a physically motivated constraint within the NN itself making this model extremely versatile, especially in the modelling of different surface charge states. The NN is validated considering the redistribution of electronic charge density within a graphene based electrode in contact with aqueous electrolyte solution, a system highly relevant to the development of next generation low-cost supercapacitors. We compare the performances of our NN/MD model against Quantum Mechanics/Molecular dynamics simulations where we obtain a most satisfactory agreement.

The first reports of Machine Learning (ML) in computational materials modelling emerged close to three decades ago[1], yet only very recently has their presence in this field become ubiquitous, in particular in the form of *Supervised Learning* (SL)[2–4]. SL encompasses a group of methodologies with the same principal philosophy: given a set of observations in the form of input and output data, the goal is to determine a model that can make accurate output predictions given an arbitrary input. Examples of SL within materials modelling include Gaussian Approximated Potentials (GAP)[5–7] that use Gaussian random processes to predict atomistic Potential Energy Surfaces (PES)[8–10], Kriging regression[11,12], which is used in the geometrical optimization of molecules[13] and PES prediction[14–16], kernel-ridge regression for the description of the multipole of a molecule[17,18], and Neural Networks (NN) that are also used to make predictions about the PES[8–10] and predict the difference between forces obtained from DFT and classical force fields[19]. In particular, NNs represent the one most widely utilized techniques in materials modelling owing to their versatile and broad applications; NNs have been applied to the parametrization of wave functions[20] and quantum density matrices[21] as well as applications within Quantum Monte Carlo simulations[22]. More recently, NN also found applications in Computational Fluid Dynamics Simulations[23,24]. Here, NNs are employed to model the quantum mechanical fluctuations of the electron density of a charged solid surface that lead to polarization effects at solid-liquid interfaces.

In the investigation of solid-liquid interfaces by classical Molecular Dynamics (MD) simulations, the standard practice in all-atom approaches is to assign to each species a fixed charge that is representative of its nuclear charge plus an attributable proportion of the average shared electron density. Polarisation effects that give rise to stronger attractive or repulsive interactions between non-bonded species may be treated in an average way through a parameterized non-bonded Lennard-Jones interaction potential[25,26]. However, this treatment neglects the dynamical aspect of the interface polarizability which can be important for capturing the correct physisorption and diffusion behaviour[27,28]. In strictly metallic systems, the redistribution of the electronic density, and thence surface charge, can be modelled for instance by the constant potential method[29–31]. In semiconducting and insulating materials polarizable force fields can be applied to surfaces, accounting for dynamical effects by tethering a dummy charge to polarizable atoms via a harmonic spring, thereby allowing for modulation of the atomic charge density in response to the environment.[28,32,33] In the case of graphene/electrolyte interfaces, in our previous work we observed that ions induce a long ranged redistribution of surface electron densities that can only be accurately accounted for by methods which compute directly the electronic surface density[34]. To this end we implemented an iterative Quantum Mechanics/Molecular Dynamics (QM/MD) workflow, by which the dynamics of surface-electrolyte interfaces can be modelled in a classical framework all the while including a QM description of the polarization of the surface.

In the QM/MD scheme the state of the surface polarization evolves in response to the local electrostatic potential arising from the relative positions of molecules in the liquid phase. This could be for instance the water molecule dipole and/or charges associated with a solute. At a given time-step, the specific configuration of surface charges are obtained through Mulliken population analysis of the electronic charge density obtained at the Density Functional Tight Binding (DFTB) level of theory. In order to avoid very large-scale quantum mechanical calculations, only the surface atoms are treated by DFTB, and the specific arrangement of the electrolyte atoms enters into the calculation as a field of point charges. The DFTB surface atom populations are translated to a set of atomic charges and included as parameters in the classical MD force field (FF). Iteration of this procedure for many time steps ensures that there is feedback between the classically determined positions of the electrolyte atoms and the QM derived surface charges.

Within this framework, the need of the DFTB calculations represents the bottleneck for the simulation time, and a trade-off between the accuracy and practical viability of the simulation must be established for the feedback between the quantum and classical model.

High frequency updates of the surface charge would improve the sampling of the electronic potential energy surface and reduce the time lag between the QM and MD calculations, but this comes at the cost of a slow down in the simulation time. Supervised Learning of the QM polarizability, in particular using NNs, represents a novel avenue by which we can avoid the computationally expensive QM calculations, instead calling upon a trained model *in-situ* to retain the dynamical description of the polarizability of the surface. Along these lines, this work, introduces a NN model for the on-the-fly prediction of the surface atomic partial charges, within a classical force field (FF), that accounts for the evolving polarizability of the surface. The NN, which is trained over the QM calculations is then fully integrated into a ML/MD workflow replacing the QM calculations, still effectively reaching the same goal of obtaining an improved FF which is not constrained by fixed point charges.

It is worth highlighting, our approach includes substantial deviations from the standard application of NN models to MD, more specifically: (**i**) We propose a multi-output neural network scheme[35], where a single NN gives for each prediction the instantaneous value of the charge on *each* atom of the surface, (**ii**) We introduce a formal constraint in the generation of the NN model, to link the model to the physics of the system where the total surface charge must be preserved. This, in turn, is done by modifying the Loss Function (LF) used to train the model.

Finally, (**iii**) we simplify the problem from a standard regression one, where the prediction involves continuous intervals (i.e., numbers in $\mathbb{R}$), to a problem similar to a classification one, where we are dividing our output in a finite number of classes (i.e., we are dividing the range of values assumed by the charges in discrete intervals) and predicting in which class each charge falls instead. Although classification problems find use outside of computational materials science, they received little attention within the community. Here we show that they increase the flexibility and performances of the ML models particularly for the problem at hand (i.e. simulating a solid/liquid interface). We validate the framework simulating the interfacial properties of an electrified graphene/electrolyte interface.

The system considered here is a charged semi-infinite graphene electrode in contact with a 1M NaCl electrolyte solution. The electrode is comprised of $N_C = 336$ carbon atoms and carries an excess charge of 4 e. The electrolyte solution has 2065 water molecules and 90 and 86 fully dissociated Na^+ and Cl^- ions respectively. A sketch of this system is presented in fig. 1. It should be noted that the excess of Na ions balances the charge of the electrode, preventing problems with the computation of long-ranged electrostatic interactions in the MD step. In our previous work, (see[34]) we observed that there exist a finite amount of charge for which any two charges differing by this value are not seen as different by the system. This observation, which will be made more quantitative in the next section, will be essential for the work developed here.

The work is organized as follows: we first describe the Neural Network architecture, highlighting the novelties introduced in this paper, namely the use a multi-output NN trained with a modified loss function, and the transformation of the problem from a pure regression to a classification one. Then, we describe the setup of the simulations used to derive the training points needed to train our NN and the set up of the MD simulations that use the distribution of carbon atom charges predicted by the NN. We then show some results related to the performance of the NN and the accuracy of the NN/MD calculations. The conclusions close the manuscript.

1 Network Structure

The first important novelty added in this work stems in the way we simplify the problem by not considering it only a pure regression but a regression/classification one. This simplification is allowed by the physics of the system and the fact that the simulation results are insensitive to small changes[34] ($\epsilon = 0.015\, e$) in the carbon partial charges within a classical force field. By looking at the distribution of the charges, dividing them into bins of finite size and assigning a label to each of them, we can ask our NN to predict only the bin in which the particular charge falls rather than the actual value of the charge. That means that we translated a pure regression problem into a labelling (classification) problem, even though a standard regression seems a natural choice given the continuity of the value of the charge. Thus the problem becomes to find the correct label for each surface carbon atom (i.e. correct class in the distribution of charges) given a specific electrolyte configuration, which, in this set up, represents the input of our NN. The reduction of the cardinality of the space of the outputs from infinite to a series of finite discrete values, is a well-known procedure in the ML community. This reduction is called quantization (as in "quantize" a data set from continuous to discrete quantities) "bucketization", or "data binning". We prefer to avoid the term quantization here, since this notion is not related to quantum mechanics concepts. The use of data binning presents some advantages in the creation of the NN, namely the NN does not waste resources trying to learn all the finest details of the outputs which are physically irrelevant, as the system cannot discriminate between two charges below a certain threshold, focusing on the more essential task to reproduce the correct distribution of the charges on the graphene layer, given a particular configuration of the electrolyte. One of the disadvantages of the binning is usually the fact that there is no *a priori* prescription on how to select the bins. In our work, we do not have such a problem. We are guided by the physics, and the criteria we choose is obtained by the true distribution of the charges and the physical threshold ϵ.

Once the label is obtained, it can be mapped back to the corresponding charge. When a given charge is assigned to one of the bins it is replaced by the median value of that bin. Therefore, the size of the bin, b, must be chosen smaller than ϵ. However, we require a stricter condition on b, in particular we require that $b < \epsilon/2$. The reason lies in the discretization performed when we assign each charge to the bins, which is explained in detail in section 2.3. If $b < \epsilon/2$ then a label prediction which is close enough to the real one can be still considered correct, as it will be shown in the next section. In this work we use a value of $b = 0.007\,e$.

The basic architecture of a NN, (see fig. 1) is represented by a set on neurons in several layers; each neuron within a layer is connected to all of the neurons in the subsequent layer. The first and last layers are special ones, the first layer represents the input layer, and the number on neurons here is equal to the number of features of the problem. The last layer is the output layer and in our case is composed of several outputs[36].

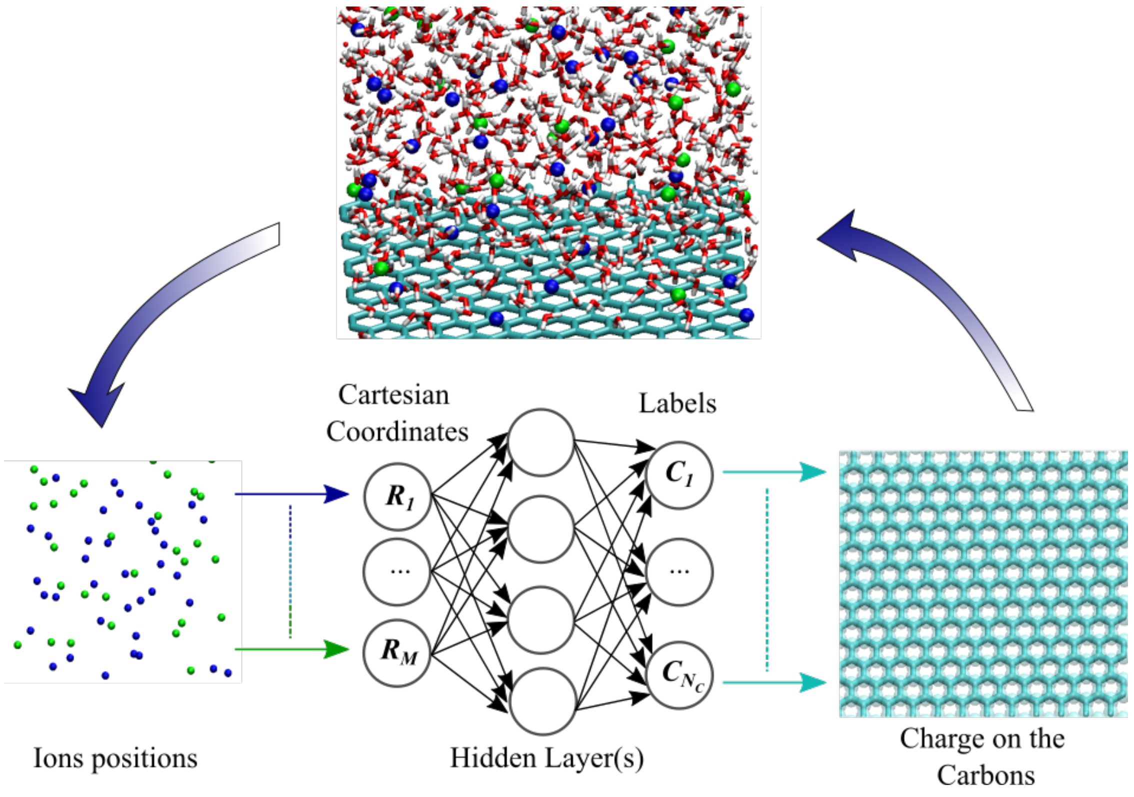


Figure 1: Sketch of the loop for NN/MD calculations which shows the sequence of operations included. We highlighted the Multi-output Neural Network step with M inputs and N_C outputs, along with the definitions of the inputs extracted from the MD simulation (the position of the ions) and outputs entering into the MD simulation (i.e. the charge on the carbons).

The output of the j-th neuron in the k-th layer, $y_j^{(k)}$, depends on the output of the previous layer $k-1$ and can be written as:

$$y_j^{(k)} = f\left(b^{(k-1)} + \sum_{i=1}^{m^{(k-1)}} w_{i,j}^k y_i^{(k-1)}\right) \tag{1}$$

where the superscript $(k-1)$ refers to objects in the $k-1$-th layer, $m^{(k-1)}$ is the number of neurons in the layer, $w^k_{i,j}$ is the weight associated to the i-th neuron, $b^{(k-1)}$ is the bias and f is the activation function. If $y^{(k)}_j$ is the output of the last layer k runs over all the different outputs.

In order to obtain a NN model, the weights $w^k_{i,j}$ must be optimized for each neuron in each layer, in practice this means minimising a certain *Loss Function* (LF), $\mathcal{L}(\mathbf{y}^{(out)}_i, \hat{\mathbf{y}}_i)$, which measures the "distance" between prediction and true value of the property:

$$\mathbf{w}^{\text{opt}} = \underset{\mathbf{w}}{\text{argmin}}\, \mathcal{L}(\mathbf{y}^{(\text{out})}, \hat{\mathbf{y}}) \tag{2}$$

where $\mathbf{y}^{(\text{out})}$ and $\hat{\mathbf{y}}$ are the N_C-dimensional vector of the respectively true and the predicted label attached to, in the present case, the N_C carbons within the graphene layer.

By definition, the true charges computed in the QM step sum to the charge applied to the electrode. In order to enforce this constraint we include it as an extra term into the loss function appearing in eq. (2). Our new loss function reads as:

$$\mathcal{L}(\mathbf{y}^{(out)}, \hat{\mathbf{y}}) = \frac{1}{N_c}\left[\sum_{i=1}^{N_C} \left|y_i^{(out),j} - \hat{y}_i\right| + \lambda \left|\sum_{i=1}^{N_C} \mathbf{y}^{(out)} \cdot \mathbb{1} - \sum_{i=1}^{N_C} \hat{\mathbf{y}} \cdot \mathbb{1}\right|\right] \tag{3}$$

where $\mathbb{1}$ is the N_C-dimensional vector of ones, and $(\cdot)$ is a scalar product. The first absolute value is the Mean Absolute Error (MAE) loss function optimized during the NN training, and represents the magnitude of error committed on the predictions. The second absolute value represents a so-called soft constraint, i.e. a penalty over the predictions, calculated as the difference between the sum of the total predicted charges and that of the true total charge. λ is an adjustable parameter which quantifies the strength of the coupling of the loss function with the penalization term. In this work, we set a value of $\lambda = 1$ which gives satisfactorily results. The modification of the loss function presented here represents one of the possible ways to include this constraint in the NN. In particular, this method of including such constraint as an additional term of the loss function is known as "regularization"[37] and the constraint we use is a type of soft-constraint. Such constraints are defined soft as opposed to hard-constraints (such as the ones calculated using Lagrange multipliers) because they can be violated. However, any such violation results in a penalization (i.e. a positive term added to the loss function). Their overall effect is therefore a penalization of the learning of unnecessary complex models. This, in turn, is represented in our case by the fact that our NN needs to learn only over models which return a total value of the property equal to $\sum_{i=1}^{N_C} \mathbf{y}^{(out)} \cdot \mathbb{1}$.

An important part of the creation of a ML model is the selection of the inputs, or *features*. In our system, the distribution of charges on the graphene sheet depends on the configuration of the water molecules and ions in the electrolyte solution. However, the correlation among the positions of the water molecules and the ions during the simulation, strongly implies that we do not need to include all the molecules in the creation of the *features*. In this work, we show that using the ions configurations is enough to obtain good descriptors for the training of the NN. This last fact represents a key observation for the generation of NN models for MD simulations and here we argue that the amount of information needed to create good ML models can be reduced with respect to the naive choice of considering everything within the system.

Now, we need to explicitly describe the *features* to be used in the NN model. Generally, Cartesian coordinates are not considered good candidates for ML models in MD. The fact that they lack some essential symmetries, i.e. they are neither translationally nor rotationally invariant, or invariant to an exchange of identical atoms, is generally a problem for ML model generation. The issue originates from the fact that NNs consider two input geometries which differ only by a rotation or translation of the system as being unique. In literature, different methods have been proposed to take into account these symmetries[38–42].

However, in the present case, the use of absolute Cartesian coordinates as input does not suffer of the problems outlined. In our work, the graphene carbons are fixed in space throughout the simulation. If two configurations differ by a translation of any ion within the simulation box, they are different with respect to the fixed graphene interface. Therefore, they effectively represents different configurations [1]. For this same reason, namely the graphene sheet is fixed in space, we do not need to consider the permutation of the carbons on the sheet. As long as the ordering of the carbons in the simulation is consistent with the training of the NN, the predictions will not need to consider the permutation of the carbon on the surface.

Another disadvantage of cartesian coordinates for the input features resides in the lack of transferability of the model with respect to the molarity of the solution (i.e., the number of ions in the system). However, as we discussed, the model we are presenting here is not relying on the use of cartesian coordinates, which are just a convenient and straightforward way to present the main novelty of this work. These considerations encompass the transferability with respect to different systems. If a different electrolyte solution is considered (i.e., different ions in water), then a different network will be needed. However, the type of problem we are considering suggests that a possible route would be to train the NN with different electrolyte solutions, to obtain a single NN able to describe different systems. This last consideration is however outside the scope of this paper and will be considered for future publications.

In terms of the output, the architecture we are proposing fixes the number of carbons of the solid surface. However, this latter fact does not represents a major inconvenience. Once a sufficiently large interface is considered, the behaviour of the electrolyte is not affected by the surface size.

The set up for the inclusion of the NN into the MD calculations is similar to the one followed for the QM/MM workflow (see also[34]) with the only difference being the replacement of the DFTB calculations for the prediction of the surface charges with the NN.

Every 5 ps the ions coordinates are extracted from the trajectory and transformed into the input configuration for the NN model, from which the new charges are predicted. A sketch of the loop of the NN/MD calculation is reported in fig. 1. In practice, this functionality is implemented as a set of drivers which couples TensorFlow[43] with Gromacs[44]. By maintaing

[1]A translation symmetry along the direction perpendicular to the plane of the electrode exists but it's not considered here since all the configurations are obtained with respect the same position of the electrode.

these as a set of flexible python drivers of third party software the framework is rigidly transferable across multiple MD software that can be chosen to suit the users needs.

2 Computational Details

2.1 QM/MD Polarized Graphene layer

We present a brief description of the simulation we used to obtain the training points for the construction of the neural network, for more details we refer to our original work[34]. The system under investigation is a charged semi-infinite graphene electrode in contact with a 1M NaCl electrolyte solution. The electrode is composed by 336 carbon atoms and carries an excess charge of 4 e. The electrolyte solution has 2065 water molecules and 90 and 86 fully dissociated Na^+ and Cl^- ions respectively. It should be noted that the excess of Na ions balances the charge of the electrode, preventing problems with the computation of long-ranged electrostatic interactions in the MD step. The dimensions of the simulation box are approximately $3 \times 3 \times 16$ nm^3, where in the system non-periodic direction (orthogonal to the electrode plane) there is an 8 nm slab of electrolyte plus a further 8 nm of vacuum separating periodic images. The configuration we considered for our system mitigates the emergence of dipole interactions across the cell. For the DFTb calculations we performed, we determined that 6 nm of electrolyte is sufficient to screen the electric field arising from the electric double layer. For the classical molecular dynamics simulations we use the Gromacs[44] software suite version 2018.4. The surface polarization evolves in response to the local electrostatic potential created by the water molecules and ions in solution, but it represents a quantum mechanical property of the surface electrons. In order to capture the redistribution of the electron density we iteratively couple density functional tight binding simulations of the graphene surface to the classical molecular dynamics trajectory. In practice we convert the coordinates of the electrolyte atoms (Hydrogen , Oxygen, Sodium and Chloride) from a snapshot of the classical trajectory into a set of point charges; the magnitude of the charge is taken from the classical force-field. The point charges form the background electrostatic potential for DFTB simulation of the graphene surface. The DFTB simulation in turn gives rise to a distribution of the surface electron density in response to the position of the electrolyte atoms. From the distribution of the electron density we estimate the atomic charges through mulliken populations of the atomic orbitals. These atomic charges define the charges used in the classical force field for the generation of the future configuration. For the aqueous graphene interfaces we find that a coupling time of 5 ps between quantum mechanical feedback is sufficient for keeping the error in the atomic charges below 0.015 e.

From these calculations we obtained 30000 different configurations saved every 1 ps to reduce the correlation among the different geometries.

2.2 Neural Network with Tensorflow

The NN model was obtained using Tensorflow/Keras library v. 2.3.1. Each of the 30000 configuration includes the positions and charges of the ions and the charges on the graphene layer, as input and output features, respectively. Therefore, the number of input features of each configurations is 704, calculated as: (number of Cl^- + number of Na^+) times four, for the three spatial coordinates and the charge of each ion. As we discussed in section 1, in our setup the neural network is trained using density functional tight binding calculations where each of the electrolyte atoms (O, H, Na, Cl) are included as point charges. This means that the resultant surface polarization seen by the neural network is polarized according to the specific configuration of the H_2O molecules. In this way, the contribution of the water molecules to the surface polarization is implicitly included during the fitting procedure allowing the use of a simplified set of input features (namely, the spatial positions of ions *only*). In turn, even if this strong correlation may not be an universal behaviour which can be easily translated to any kind of systems, the agreement of the results with the target calculations we obtained (see section 3) is an indication that such correlation can be safely assumed in this case.

We randomly split the total set of configurations in a training set composed by 27000 configurations and the holdout/test set with the remaining ones. Moreover, we use 10-fold cross validation, by dividing the training set into 10 subsets and repeating the model training 10 times, with a single subset as holdout/test set.

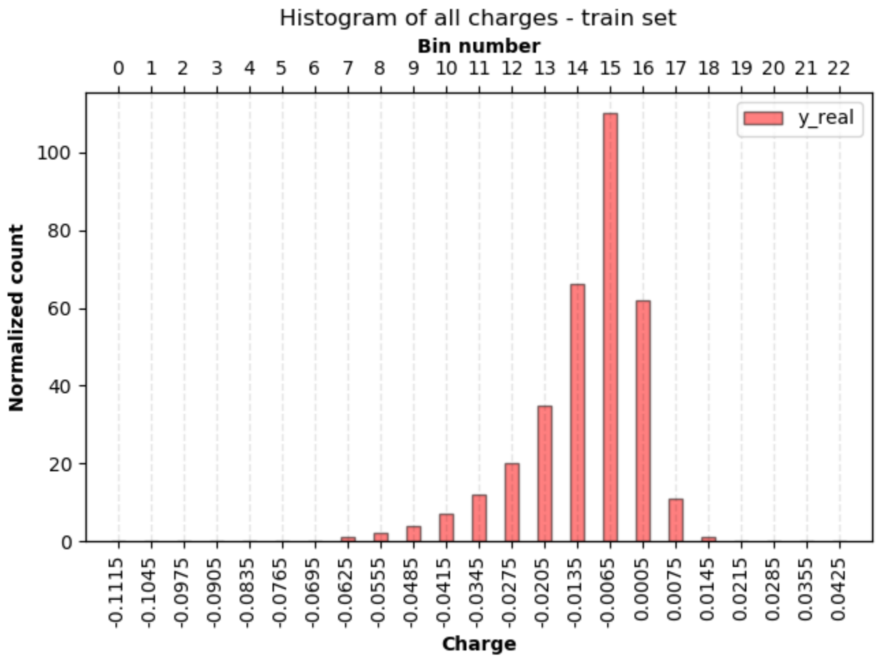


Figure 2: Histogram of the distribution of charges on the carbon atoms of the electrode obtained from 27000 training geometries. The histogram is normalized in the sense that the sum of all the bins gives the total number of carbons atoms considered on the electrode.

From the 27000 configurations in the training set, we built the histogram showing the relative frequency of the charges which is reported in fig. 2. The spacing of the bins of this histogram was chosen to be $s_b = 0.007e$. The choice of the bin spacing has some important consequences for our work and we will describe it in more detail in the next section (see section 2.3). The charge of the k-th carbon, q_k, was then assigned to n_b^k-th bin of the histogram, according to:

$$n_b^k = \left\lceil \frac{q_k - q_{\text{min}}}{q_{\text{max}} - q_{\text{min}}} \right\rceil \tag{4}$$

where q_{max} and q_{min} are determined by the smallest and largest charges encountered in the 27000 points of the training set.

Multi-output/multi-target classification is not directly supported by NNs. Therefore, we built a multi-output regression NN and explicitly round the floating point predictions to integers, i.e. the discrete charge bin numbers. In turn, these bin numbers can be mapped to the actual charge values.

The NN is a typical multilayer perceptron (MLP) with four fully-connected dense hidden layers of 1024 neurons with *ReLU* activation which represents a good compromise between speed and robustness of the model. The custom loss function that combines the normal mean absolute error loss and the loss that penalizes the differences between the sum of real and predicted values is depicted in Listing 1.

The training is performed with the Adam() optimizer with initial learning rate lr=2e-4. Moreover, the training is automatically stopped with an early stopping mechanism if no improvement of the validation loss has been observed after 100 epochs.

```
def custom_loss(y_true, y_pred):
  # MAE loss
  err = K.mean(K.abs(y_true-y_pred), axis=-1)
  # Penalization of differences between sum(y_true) and sum(y_pred)
  fac = 1.0/336.0
  constraint = fac*K.abs(K.sum(y_pred, axis=-1)-K.sum(y_true, axis=-1))

  return(err+constraint)
```

Listing 1: Code for the definition of the Custom Loss Function defined in Eq. 3, here we assumed $\lambda = 1$.

2.3 Bin definition

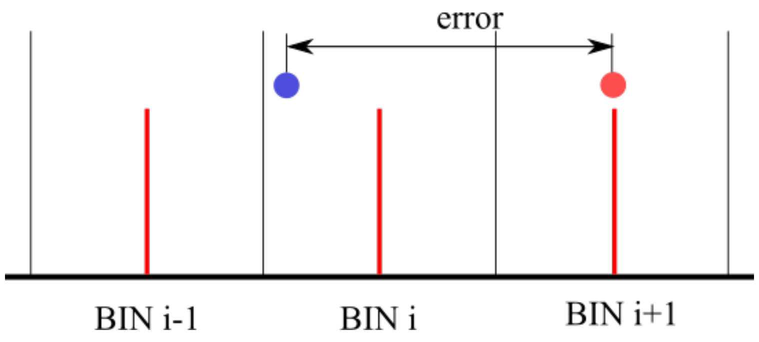


Figure 3: Sketch of the assignment of the charges into the bins. Blue dot is the real charge, which is replaced with the value of the charge assigned to the bin i (represented by the red line in bin i). Red dot is the predicted charge which is predicted in the wrong bin $i+1$. if the spacing between bins is greater than half of the threshold (see Sec. I of the main paper), then the prediction is not correct. By choosing a bin size smaller than half of the threshold, the prediction in the bin $i+1$ is still correct .

Once a given charge is assigned to one of the bins, it is replaced by the median value of that bin. In fig. 3 we show one possible scenario for the prediction of the charges. The real charge is represented by the blue dot, and it can be anywhere in the region bounded by the upper and lower extrema of the bin. This latter fact puts a constraint on the maximum size of the bin, which has to be such that $b < \epsilon$.

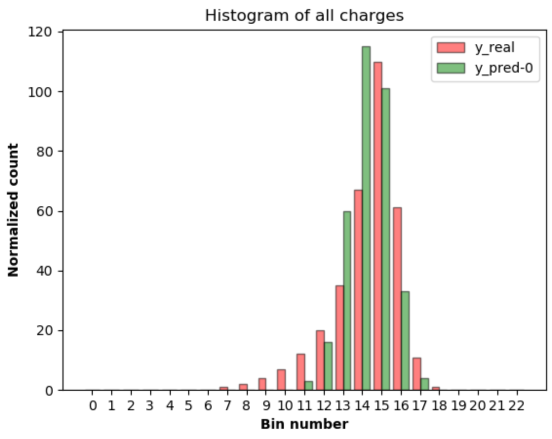
If the predicted value, represented by the red dot in fig. 3, is in the bin $i+1$, then the median value for the charge of the bin $i+1$ is assigned. If the size of the bin is ϵ, the difference between the real charge (the blue dot) and the predicted charge (the red dot) is greater than ϵ and the prediction is wrong. If we use a spacing between the bins such that $b < e/2$ we can assume that the wrongly predicted label, with a difference of ± 1 from the real one, is actually correct.

3 Results

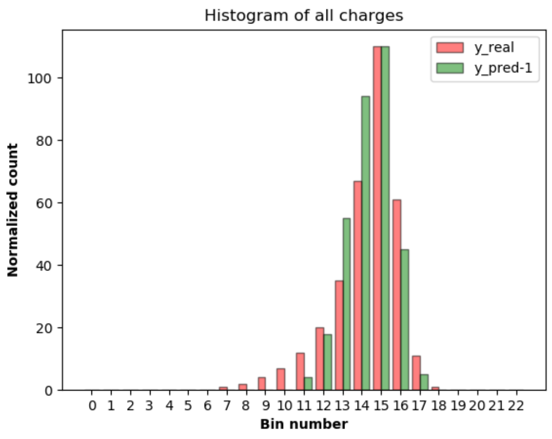
In this section we will start by showing the performances of the NN model on a prediction set composed of 3000 electrolyte configurations and the resultant carbon charges computed by DFTB simulations (to which we will refer as the “real” charges). We then conclude by reporting the results of the fully integrated NN/MD simulation and comparing various properties with those obtained from an analogous QM/MD trajectory.

In Figure 4 we plot histograms that compare the distributions of the real and predicted charges. Leveraging that *i*) the differences in the charges which are smaller than ϵ are not seen as different and *ii*) the size of the bin, b, is such that $b < \epsilon/2$, it follows that if the NN predicts a label for a charge which is ± 1 away from its correct one, it can be assigned back to its correct label. By filtering out the results in fig. 4a, which represents the distribution as obtained by the NN model, using this consideration, we obtain the histogram shown in fig. 4b. Our model is able to correctly capture the behaviour of the system in terms of identification of the most important classes, which in turn, represent the most likely observed charges. However, our model yields a lower likelihood that charges on the left tail of the distribution will be observed. These charges are those appearing with the least frequency during the time evolution of the electrolyte configurations, which are therefore likely to be under-represented using the random sampling we employed to construct the whole data set. An improved sampling of the training set may help in reducing this effect (e.g.[45]), but as we will show next, under representation of these labels does not have a noticeable effect on the NN/ML simulation results.

Figure 5 reports the sum of the charges, for each frame, of the predicted set, with and without the constraint applied in the loss function. This serves to highlight the importance of the constraint since without it the sum of the predicted charges has a mean value different from the one set in the classical step (4.0 e for the system considered here). If the NN does not conserve the electrode charge, then firstly, the modelled system is fundamentally different from the real system and secondly, on a more technical note this can lead to instabilities in the evaluation of the Ewald summation during the classical MD step, where the overall electrolyte charge no longer counter balances the electrode. In fact, our results suggest that the inclusion of a physically motivated term within the loss function can lead to better



(a)



(b)

Figure 4: Histogram of the true charges as calculated by the DFTB (in red) and comparison with their NN predicted values (in green). On the left panel we reported the charges as they are predicted from the NN, on the right panel we report the filtered charges using the threshold defined in the main text.

models generally. One thing to highlight here is that, while the average of the total charge shown in fig. 5 has the correct mean, its instantaneous value can be different from 4.0 e which is needed to ensure the neutrality of the system. In order to avoid this effect, in this work we enforced the neutrality by calculating the difference between the instantaneous total charge after each prediction and dividing it evenly among all the carbons on the surface. This procedure is not going to modify the system, since the charge added to each carbon on the graphene layer is well below the treshold ϵ [2].

We have shown in fig. 4 and fig. 5 the performance of the predictions in terms of the relative frequency of the charges compared with the real ones.

On top of their histograms, we can also consider the real space distributions of the predicted labels in comparison with the true charges, since this will give rise to the dynamical feedback with the electrolyte during the NN/MD loop. In particular, this difference between the QM and NN charges should be minimal in order to avoid nonphysical charge (de)localization.

[2]Let us assume an error as large as 1 e, then $1/336 \approx 0.003$ e which is a quantity well below ϵ.

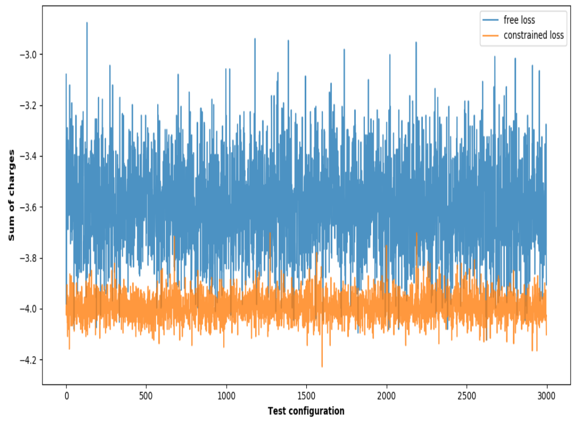


Figure 5: Sum of charges for the test configurations

The comparison between the true and predicted charges has been carried out on the test set by calculating the difference between the value of the real charges and the prediction of the NN model on the same configuration, which we report in fig. 6. If the error on the charge is smaller than the threshold of 0.015 than an error of zero is assigned to that particular carbon. We observe that given this constraint the difference between the real and predicted charges is zero for the majority of C atoms across all frames. Moreover, where individual C atoms take a value different from zero, the prediction appears to be isolated in space and across different frames. As a consequence, the resultant polarization of the sheet is not affected and the contribution of these larger deviations is averaged out over the course of even several tens of ps. As observed for fig. 4, the NN has a slight bias towards highest labels which can be possibly mitigated by a more accurate generation of the data set points, but overall the qualitative behavior is similar to the one observed in QM calculations with regions with a larger (negative) charge, regions mostly neutral and very few positive charges.

The distribution of the predicted charged gives the overall behaviour of the predictions, but does not give any indication on the error committed in each prediction. The most straightforward evaluation of the performances of any NN is the prediction error with respect to the charges on the prediction set. As shown in fig. 7, where we report an S-curve showing the percentile on the y-axis and the absolute error on the x-axis. Each point gives on the

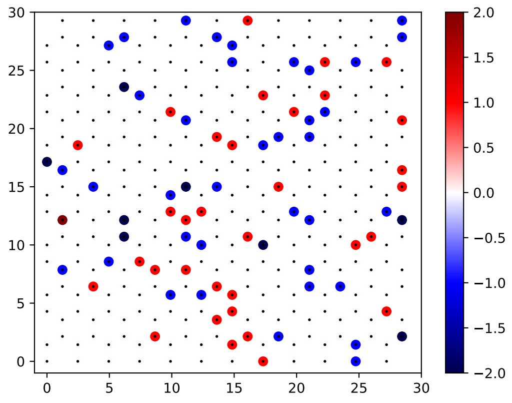


Figure 6: A two dimensional plot of the difference between the QM and NN distribution of charges on the Carbon atoms in the graphene sheet for a randomly selected snapshot, a movie over the trajectory is available as part of the SI. The legend is given in units of the threshold ϵ.

y-axis the percentage of the carbons in the predictions set with error lower than the one marked by the x position. The error is given in terms of the distance of the predicted label from the true one. A distance of zero means the label was correctly predicted. The S-curve for the predictions on the charges on the graphene layer is plotted in fig. 7. Even though each prediction gives all the charges on the graphene layer at the same time, we consider for fig. 7 each charges separately, i.e. the picture is showing the errors committed on a single charge. In this figure we also included a black vertical line at $0.015\,e$. From fig. 7 it results that with the $0.015\,e$ threshold we can consider correct almost 87% of the charges predicted. Naturally, the threshold we used has still to be tested in a simulation, where we can confirm that such an approximation is enough to obtain reliable results from the MD simulations. Figure 8 shows the normalized density of water, $Na^+$and Cl^- across the simulation box as function of the distance from the graphene layer (which, in our configuration is perpendicular to the z-direction and is located at $z = 0$). As expected, the density of the Cl^- ion is smaller than the Na^+near the surface since the graphene layer is negatively charged. One thing we can notice from fig. 8 by comparing the relative height of the first peaks for water and sodium (comparable to the first solvation shell for the graphene layer) is that the relative height of

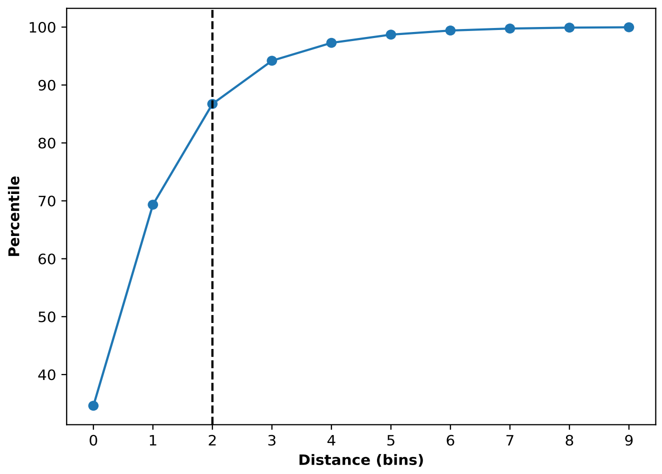


Figure 7: S curves showing the distance of the predicted bin with respect the correct value. The vertical black dashed line represents the threshold ϵ.

the peaks is preserved in NN/MD. The distribution of the chlorine shows a better agreement with QM/MD calculations than the sodium. This fact could be due to the fact that chlorine being, on average, repelled from the interface is less sensitive to the small differences between the QM and NN description of the graphene layer.

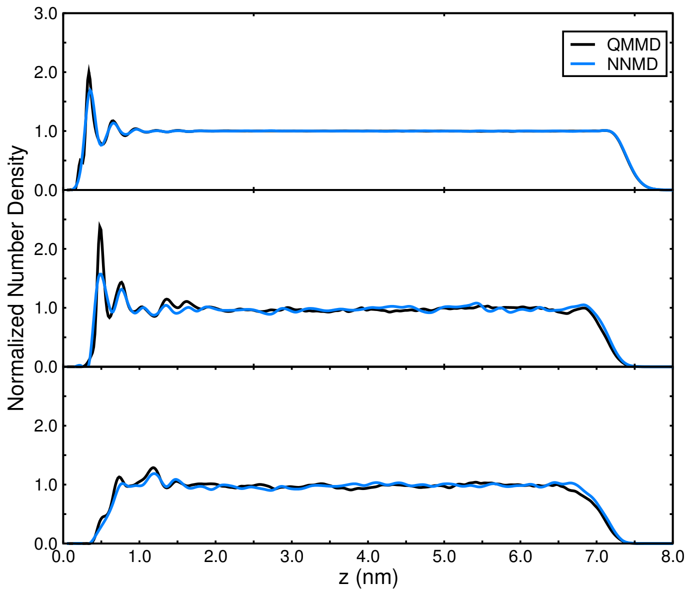


Figure 8: From top to bottom: normalized density of water, Na^+ and Cl^- as function of the distance from the graphene layer for QM/MD, NN/MD calculations.

These results show the potential of this different paradigm for NN for MD simulations. In particular, we showed how the classification problem can be used in the generation of NN for MD calculations instead of the more complicate regression one. We showed that Cartesian coordinates can be used as features, giving good results, even though other feature selection may improve the predictions as well as a more clever choice of the training set geometries. In terms of speed-up of the calculations, the time needed for a NN/MD simulation is of the same order of magnitude of a standard MD simulation, which is a huge computational advantage compared with any other procedure to include surface polarizability. However, the model presented here suffers of a not fully integration with the MD code, which surely represents the next step in the development of NN/MD models.

4 Acknowledgment

The authors thank the European Union's Horizon 2020 research and innovation programme project VIMMP under grant agreement no. 760907. The authors would like to acknowledge the assistance given by Research IT and the use of The HPC Pool funded by the Research Lifecycle Programme at The University of Manchester.

5 Supplementary Information

The video included represents a two dimensional plot of the difference between the QM and NN distribution of charges on the Carbon atoms in the graphene sheet for a 10 ns trajectory. The legend is given in units of the threshold ϵ.

This information is available free of charge via the Internet at http://pubs.acs.org.

References

(1) Blank, T. B.; Brown, S. D.; Calhoun, A. W.; Doren, D. J. Neural Network Models of Potential Energy Surfaces. *J. Chem. Phys.* **1995**, *103*, 4129–4137.

(2) Goh, G. B.; Hodas, N. O.; Vishnu, A. Deep Learning for Computational Chemistry. *J. Comput. Chem.* **2017**, *38*, 1291–1307.

(3) Noé, F.; Tkatchenko, A.; Müller, K.-R.; Clementi, C. Machine Learning for Molecular Simulation. *Annu. Rev. Phys. Chem.* **2020**, *71*, 361–390.

(4) Zhang, J.; Lei, Y.-K.; Zhang, Z.; Chang, J.; Li, M.; Han, X.; Yang, L.; Yang, Y. I.; Gao, Y. Q. A Perspective on Deep Learning for Molecular Modeling and Simulations. *J. Phys. Chem. A* **2020**, *124*, 6745–6763.

(5) Bartók, A. P.; Payne, M. C.; Kondor, R.; Csányi, G. Gaussian Approximation Potentials: The Accuracy of Quantum Mechanics, without the Electrons. *Phys. Rev. Lett.* **2010**, *104*, 136403.

(6) Bartók, A. P.; Csányi, G. Gaussian Approximation Potentials: A Brief Tutorial Introduction. *Int. J. Quantum Chem.* **2015**, *115*, 1051–1057.

(7) Boussaidi, M. A.; Ren, O.; Voytsekhovsky, D.; Manzhos, S. Random Sampling High Dimensional Model Representation Gaussian Process Regression (RS-HDMR-GPR) for Multivariate Function Representation: Application to Molecular Potential Energy Surfaces. *J. Phys. Chem. A* **2020**, *124*, 7598–7607.

(8) Behler, J. Neural Network Potential-Energy Surfaces in Chemistry: a Tool for Large-Scale Simulations. *Phys. Chem. Chem. Phys.* **2011**, *13*, 17930–17955.

(9) Behler, J.; Parrinello, M. Generalized Neural-Network Representation of High-Dimensional Potential-Energy Surfaces. *Phys. Rev. Lett.* **2007**, *98*, 146401.

(10) Schmitz, G.; Godtliebsen, I. H.; Christiansen, O. Machine Learning for Potential Energy Surfaces: An Extensive Database and Assessment of Methods. *J. Chem. Phys.* **2019**, *150*, 244113.

(11) Di Pasquale, N.; Davie, S. J.; Popelier, P. L. A. Optimization Algorithms in Optimal Predictions of Atomistic Properties by Kriging. *J. Chem. Theory Comput.* **2016**, *12*, 1499–1513.

(12) Di Pasquale, N.; Bane, M.; Davie, S. J.; Popelier, P. L. A. FEREBUS: Highly Parallelized Engine for Kriging Training. *J. Comput. Chem.* **2016**, *37*, 2606–2616.

(13) Zielinski, F.; Maxwell, P. I.; Fletcher, T. L.; Davie, S. J.; Di Pasquale, N.; Cardamone, S.; Mills, M. J. L.; Popelier, P. L. A. Geometry Optimization with Machine Trained Topological Atoms. *Sci. Rep.* **2017**, *7*, 12817.

(14) Davie, S. J.; Di Pasquale, N.; Popelier, P. L. Incorporation of Local Structure into Kriging Models for the Prediction of Atomistic Properties in the Water Decamer. *J. Comput. Chem.* **2016**, *37*, 2409–2422.

(15) Maxwell, P.; di Pasquale, N.; Cardamone, S.; Popelier, P. L. The Prediction of Topologically Partitioned Intra-Atomic and Inter-Atomic Energies by the Machine Learning Method Kriging. *Theor. Chem. Acc.* **2016**, *135*, 195.

(16) Di Pasquale, N.; Davie, S. J.; Popelier, P. L. The Accuracy of *Ab Initio* Calculations without *Ab Initio* Calculations for Charged Systems: Kriging Predictions of Atomistic Properties for Ions in Aqueous Solutions. *J. Chem. Phys.* **2018**, *148*, 241724.

(17) Bereau, T.; Andrienko, D.; Von Lilienfeld, O. A. Transferable Atomic Multipole Machine Learning Models for Small Organic Molecules. *J. Chem. Theory Comput.* **2015**, *11*, 3225–3233.

(18) Scherer, C.; Scheid, R.; Andrienko, D.; Bereau, T. Kernel-Based Machine Learning for Efficient Simulations of Molecular Liquids. *J. Chem. Theory Comput.* **2020**, *16*, 3194–3204.

(19) Pattnaik, P.; Raghunathan, S.; Kalluri, T.; Bhimalapuram, P.; Jawahar, C. V.; Priyakumar, U. D. Machine Learning for Accurate Force Calculations in Molecular Dynamics Simulations. *J. Phys. Chem. A* **2020**, *124*, 6954–6967.

(20) Carleo, G.; Troyer, M. Solving the Quantum Many-Body Problem with Artificial Neural Networks. *Science* **2017**, *355*, 602–606.

(21) Hartmann, M. J.; Carleo, G. Neural-Network Approach to Dissipative Quantum Many-Body Dynamics. *Phys. Rev. Lett.* **2019**, *122*, 250502.

(22) Fournier, R.; Wang, L.; Yazyev, O. V.; Wu, Q. Artificial Neural Network Approach to the Analytic Continuation Problem. *Phys. Rev. Lett.* **2020**, *124*, 056401.

(23) Marcato, A.; Boccardo, G.; Marchisio, D. A Computational Workflow to Study Particle Transport and Filtration in Porous Media: Coupling CFD and Deep Learning. *Chem. Eng. J.* **2021**, *417*, 128936.

(24) Saeedan, M.; Nazar, A. R. S.; Abbasi, Y.; Karimi, R. CFD Investigation and Neutral Network Modeling of Heat Transfer and Pressure Drop of Nanofluids in Double Pipe Helically Baffled Heat Exchanger with a 3-D Fined Tube. *Appl. Therm. Eng.* **2016**, *100*, 721–729.

(25) Williams, C. D.; Dix, J.; Troisi, A.; Carbone, P. Effective Polarization in Pairwise Potentials at the Graphene–Electrolyte Interface. *J. Phys. Chem. Lett.* **2017**, *8*, 703.

(26) Dočkal, J.; Moučka, F.; Lísal, M. Molecular Dynamics of Graphene–Electrolyte Interface: Interfacial Solution Structure and Molecular Diffusion. *J Phys. Chem. C* **2019**, *123*, 26379.

(27) Misra, R. P.; Blankschtein, D. Uncovering a Universal Molecular Mechanism of Salt Ion Adsorption at Solid/Water Interfaces. *Langmuir* **2021**, *37*, 722–733.

(28) Misra, R. P.; Blankschtein, D. Ion Adsorption at Solid/Water Interfaces: Establishing the Coupled Nature of Ion–Solid and Water–Solid Interactions. *J. Phys. Chem. C* **2021**, 10480acs.jpcc.0c09855.

(29) Merlet, C.; Péan, C.; Rotenberg, B.; Madden, P. A.; Simon, P.; Salanne, M. Simulating Supercapacitors: Can We Model Electrodes As Constant Charge Surfaces? *J. Phys. Chem. Lett.* **2013**, *4*, 264–268.

(30) Wang, Z.; Yang, Y.; Olmsted, D. L.; Asta, M.; Laird, B. B. Evaluation of the Constant Potential Method in Simulating Electric Double-Layer Capacitors. *J. Chem. Phys.* **2014**, *141*, 184102.

(31) Scalfi, L.; Limmer, D. T.; Coretti, A.; Bonella, S.; Madden, P. A.; Salanne, M.; Rotenberg, B. Charge Fluctuations from Molecular Simulations in the Constant-Potential Ensemble. *Phys. Chem. Chem. Phys.* **2020**, *22*, 10480–10489.

(32) Misra, R. P.; Blankschtein, D. Insights on the Role of Many-Body Polarization Effects in the Wetting of Graphitic Surfaces by Water. *J. Phys. Chem. C* **2017**, *121*, 14.

(33) Pykal, M.; Langer, M.; Blahová Prudilová, B.; Banáš, P.; Otyepka, M. Ion Interactions Across Graphene in Electrolyte Aqueous Solutions. *J. Phys. Chem. C* **2019**, *123*, 9799.

(34) Elliott, J. D.; Troisi, A.; Carbone, P. A QM/MD Coupling Method to Model the Ion-Induced Polarization of Graphene. *J. Chem. Theory Comput.* **2020**, *16*, 5253–5263.

(35) Xu, D.; Shi, Y.; Tsang, I. W.; Ong, Y.-S.; Gong, C.; Shen, X. Survey on Multi-Output Learning. *IEEE Trans. Neural Netw. Learn. Syst.* **2019**,

(36) Borchani, H.; Varando, G.; Bielza, C.; Larrañaga, P. A Survey on Multi-Output Regression. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* **2015**, *5*, 216–233.

(37) Karpatne, A.; Atluri, G.; Faghmous, J. H.; Steinbach, M.; Banerjee, A.; Ganguly, A.; Shekhar, S.; Samatova, N.; Kumar, V. Theory-Guided Data Science: A New Paradigm for Scientific Discovery from Data. *IEEE Trans. Knowl. Data En.* **2017**, *29*, 2318–2331.

(38) Davie, S. J.; Di Pasquale, N.; Popelier, P. L. A. Kriging Atomic Properties with a Variable Number of Inputs. *J. Chem. Phys.* **2016**, *145*, 104104.

(39) Ceriotti, M.; Willatt, M. J.; Csányi, G. Machine Learning of Atomic-Scale Properties Based on Physical Principles. *Handbook of Materials Modeling: Methods: Theory and Modeling* **2020**, 1911–1937.

(40) Botu, V.; Ramprasad, R. Learning Scheme to Predict Atomic Forces and Accelerate Materials Simulations. *Phys. Rev. B* **2015**, *92*, 094306.

(41) Jinnouchi, R.; Miwa, K.; Karsai, F.; Kresse, G.; Asahi, R. On-The-Fly Active Learning of Interatomic Potentials for Large-Scale Atomistic Simulations. *J. Phys. Chem. Lett.* **2020**, *11*, 6946–6955.

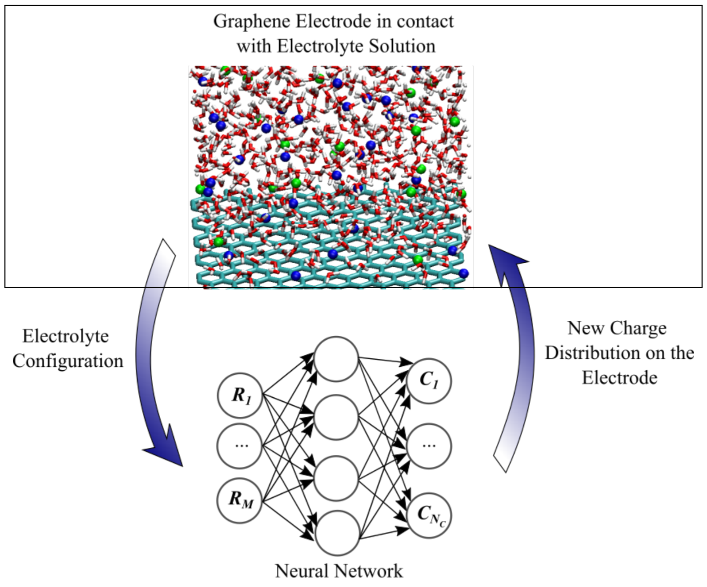
(42) Zuo, Y.; Chen, C.; Li, X.; Deng, Z.; Chen, Y.; Behler, J.; Csányi, G.; Shapeev, A. V.; Thompson, A. P.; Wood, M. A.; Ong, S. P. Performance and Cost Assessment of Machine Learning Interatomic Potentials. *J. Phys. Chem. A* **2020**, *124*, 731–745.

(43) Abadi, M.; Barham, P.; Chen, J.; Chen, Z.; Davis, A.; Dean, J.; Devin, M.; Ghemawat, S.; Irving, G.; Isard, M.; Kudlur, M.; Levenberg, J.; Monga, R.; Moore, S.; Murray, D. G.; Steiner, B.; Tucker, P.; Vasudevan, V.; Warden, P.; Wicke, M.; Yu, Y.; Zheng, X. Tensorflow: A System for Large-Scale Machine Learning. 12th {USENIX} symposium on operating systems design and implementation ({OSDI} 16). 2016; pp 265–283.

(44) Abraham, M. J.; Murtola, T.; Schulz, R.; Páll, S.; Smith, J. C.; Hess, B.; Lindahl, E. GROMACS: High Performance Molecular Simulations through Multi-Level Parallelism from Laptops to Supercomputers. *SoftwareX* **2015**, *1*, 19–25.

(45) Gastegger, M.; Behler, J.; Marquetand, P. Machine Learning Molecular Dynamics for the Simulation of Infrared Spectra. *Chem. Sci.* **2017**, *8*, 6924–6935.

Graphical TOC Entry


Graphene Electrode in contact with Electrolyte Solution
Electrolyte Configuration
New Charge Distribution on the Electrode
R_1
...
R_M
C_1
...
C_{N_c}
Neural Network