Received: 31 August 2023 | Revised: 10 November 2023 | Accepted: 11 November 2023
DOI: 10.1002/jcc.27269

RESEARCH ARTICLE

Journal of COMPUTATIONAL CHEMISTRY WILEY

Training machine learning potentials for reactive systems: A Colab tutorial on basic models

Xiaoliang Pan[1] | Ryan Snyder[2] | Jia-Ning Wang[3] | Chance Lander[1] | Carly Wickizer[1] | Richard Van[1,4] | Andrew Chesney[1] | Yuanfei Xue[3] | Yuezhi Mao[5] | Ye Mei[3,6,7] | Jingzhi Pu[2] | Yihan Shao[1]

[1]Department of Chemistry and Biochemistry, University of Oklahoma, Norman, Oklahoma, USA
[2]Department of Chemistry and Chemical Biology, Indiana University-Purdue University Indianapolis, Indianapolis, Indiana, USA
[3]State Key Laboratory of Precision Spectroscopy, School of Physics and Electronic Science, East China Normal University, Shanghai, China
[4]Laboratory of Computational Biology, National, Heart, Lung and Blood Institute, National Institutes of Health, Bethesda, Maryland, USA
[5]Department of Chemistry and Biochemistry, San Diego State University, San Diego, California, USA
[6]NYU-ECNU Center for Computational Chemistry at NYU Shanghai, Shanghai, China
[7]Collaborative Innovation Center of Extreme Optics, Shanxi University, Taiyuan, Shanxi, China

Correspondence
Xiaoliang Pan, Department of Chemistry and Biochemistry, University of Oklahoma, Norman, Oklahoma, USA.
Email: panxl@ou.edu

Ryan Snyder, Department of Chemistry and Chemical Biology, Indiana University-Purdue University Indianapolis, Indianapolis, Indiana, USA.
Email: rymsnyde@indiana.edu

Yuezhi Mao, Department of Chemistry and Biochemistry, San Diego State University, San Diego, California, USA.
Email: ymao2@sdsu.edu

Ye Mei, State Key Laboratory of Precision Spectroscopy, School of Physics and Electronic Science, East China Normal University, Shanghai, China.
Email: ymei@phy.ecnu.edu.cn

Jingzhi Pu, Department of Chemistry and Chemical Biology, Indiana University-Purdue University Indianapolis, Indianapolis, Indiana, USA.
Email: jpu@iupui.edu

Yihan Shao, Department of Chemistry and Biochemistry, University of Oklahoma, Norman, OK 73019, USA.
Email: yihan.shao@ou.edu

Funding information
National Institutes of Health; National Natural Science Foundation of China; National Science Foundation

Abstract

In the last several years, there has been a surge in the development of machine learning potential (MLP) models for describing molecular systems. We are interested in a particular area of this field — the training of system-specific MLPs for reactive systems — with the goal of using these MLPs to accelerate free energy simulations of chemical and enzyme reactions. To help new members in our labs become familiar with the basic techniques, we have put together a self-guided Colab tutorial (https://cc-ats.github.io/mlp_tutorial/), which we expect to be also useful to other young researchers in the community. Our tutorial begins with the introduction of simple feedforward neural network (FNN) and kernel-based (using Gaussian process regression, GPR) models by fitting the two-dimensional Müller-Brown potential. Subsequently, two simple descriptors are presented for extracting features of molecular systems: symmetry functions (including the ANI variant) and embedding neural networks (such as DeepPot-SE). Lastly, these features will be fed into FNN and GPR models to reproduce the energies and forces for the molecular configurations in a Claisen rearrangement reaction.

KEYWORDS
Gaussian process regression, machine learning potential, neural network, tutorial

Xiaoliang Pan, Ryan Snyder, Jia-Ning Wang, Chance Lander, and Carly Wickizer contributed equally to this work.

1 | INTRODUCTION

In recent years, there have been significant progresses in the development of machine learning potentials (MLPs) for generating high-quality potential energy surfaces for chemical systems.[1–16] In general, for a molecule with N atoms, a MLP model takes an input (Cartesian coordinates) vector $\mathbf{R} \in \mathbb{R}^{3N}$ along with the atom types, and returns the potential energy E of the system. The forces on the atoms, the negative of the derivative of the potential energy with respect to the coordinates, are calculated often through autodifferentiation. These MLPs can be categorized into two main groups based on their architecture:[8] descriptor-based models and graph neural network (GNN)-based models.

For descriptor-based models, the system's coordinates are first transformed into descriptor vectors, which must adhere to translational, rotational, and permutational symmetries. In the Behler-Parrinello neural network (BPNN)[17] and its ANI variants,[18–20] for instance, symmetric functions are used to encode the local environment of each atom into a descriptor called an atomic environment vector (AEV). In the DeepPot-SE models,[21–25] on the other hand, embedding neural networks are used to transform the coordinates into descriptors. These and other descriptors (such as the internal coordinates,[26] Coulomb matrix,[27] permutation invariant polynomial,[5,14,28,29] bag of bonds,[30] normalized inverted internuclear distances,[31] FCHL representation,[32] and weighted symmetry functions[33]) are then used as inputs to a regressor, such as a neural network or a kernel-based regressor, to predict the target molecular energy and the corresponding atomic forces.

In an alternative approach, GNN-based models treat the molecular system as a dense graph, with each atom representing a node and two-body interactions represented by edges between the nodes. Unlike descriptor-based models where the descriptors are calculated from the atomic coordinates in one pass, in GNN-based models, the description for each atom's local environment is updated iteratively through multiple rounds of refinements. Examples for this category include DTNN,[34] SchNet,[35,36] PhysNet,[37] NequIP,[38] etc.

In this tutorial on basic MLPs for reactive systems, which we prepared in the last year for training new members in our labs, we primarily focused on descriptor-based models, specifically the atom-centered symmetry functions (including the ANI variant) and the DeepPot-SE descriptors. We then employed these descriptors in combination with two types of regressors — neural network models and Gaussian process regression (GPR)[39,40] based kernel models—to train the MLPs for model systems.

This tutorial is organized as follows. In Section 2, we will briefly introduce the underlying methods for feature extraction (symmetry functions and DeepPot-SE) and for data regression (neural networks and GPR). Seven tutorial lessons will be briefly outlined in Section 3. A discussion is presented in Section 4 on the utilization and extension of these MLPs. Concluding remarks are made in Section 5.

2 | METHODS

2.1 | Feature extraction

2.1.1 | Symmetry functions

Atomic feature vectors $\{\mathbf{G}_i\}$, also known as symmetry functions, describe the organization of the environment surrounding each atom, and are usually decomposed into two-body and three-body terms. The two-body terms, which are called the radial functions following the nomenclature of Behler and Parrinello,[17] for the i^{th} atom are defined as follows:

$$G_i^1 = \sum_{j \neq i}^{N} e^{-\eta\left(R_{ij}-R_s\right)^2} f_c\left(R_{ij}\right), \tag{1}$$

summing up the contributions from all atoms other than the i^{th} atom itself. Here, f_c is a damping function of the interatomic distance R_{ij} with a cutoff R_c defined as follows:

$$f_c(R_{ij}) = \begin{cases} \frac{1}{2}\left[\cos\left(\frac{\pi R_{ij}}{R_c}\right)+1\right], & R_{ij} \leq R_c, \\ 0, & R_{ij} > R_c. \end{cases} \tag{2}$$

Note that η and R_s in Equation (1) as well as R_c in Equation (2) are all predetermined hyperparameters. The three-body terms, or the angular functions, for the i^{th} atom are defined as follows:

$$G_i^2 = 2^{1-\zeta} \sum_{\substack{j,k \neq i \\ j \neq k}}^{N} \left(1+\cos\left(\theta_{ijk}-\theta_s\right)\right)^{\zeta} e^{-\eta\left(R_{ij}^2+R_{jk}^2+R_{ik}^2\right)} f_c\left(R_{ij}\right) f_c\left(R_{jk}\right) f_c\left(R_{ik}\right) \tag{3}$$

with the j-i-k being angle

$$\theta_{ijk} = \arccos\left(\frac{\mathbf{R}_{ij} \cdot \mathbf{R}_{ik}}{R_{ij} R_{ik}}\right), \tag{4}$$

where $\mathbf{R}_{ij}$ is a vector pointing from atom i with coordinate $\mathbf{R}_j$ to atom j with coordinate $\mathbf{R}_i$, that is,

$$\mathbf{R}_{ij} = \mathbf{R}_j - \mathbf{R}_i. \tag{5}$$

Here, ζ and η are hyperparameters, and $\theta_s = 0$ or π. In the ANI[18] implementation of BPNN, the angular function is replaced with

$$G_i^2 = 2^{1-\zeta} \sum_{\substack{j,k \neq i \\ j \neq k}}^{N} \left(1+\cos\left(\theta_{ijk}-\theta_s\right)\right)^{\zeta} e^{-\eta\left(\frac{R_{ij}+R_{ik}}{2}-R_s\right)^2} f_c\left(R_{ij}\right) f_c\left(R_{ik}\right), \tag{6}$$

where R_s is the shifting hyperparameter defining the center of the Gaussian. With different combinations of the hyperparameters, a series of symmetry functions G_i^1 and G_i^2 can be defined, enhancing the capability of characterizing the inhomogeneous environment.

With the cut-off distance R_c, the BPNN potential becomes short-ranged. For systems where the long-range interaction is non-negligible, for instance for molecules in the condensed phase, the Coulomb interaction beyond the cutoff distance can still be non-negligible. For these kinds of systems, one extra feature representing the electrostatic potential embedding the atom can be appended. It can be seen that the atomic feature vectors do not depend on the absolute position of the atoms but the relative positions among all the atoms, therefore the mandatory translational and rotational invariances are satisfied. Behler and Parrinello used a fully-connected feedforward neural network for the atomic features-to-energy perception. Instead of individual neural networks for each atom, atoms of the same element share the same neural network. More generally, atoms of the same atom type share the same neural work. In other words, the neural network is not atom-wise, but element-wise or atom-type-wise. In this way, the condition of permutational invariance is also met.

2.1.2 | DeepPot-SE representation

Similar to the symmetry functions, in Deep Potential - Smooth Edition (DeepPot-SE),[21] for a system consisting of N atoms, each atom i $(1 \le i \le N)$ is first represented by its local environment matrix $\mathscr{R}^i$, that is, the relative coordinates between atom i and each of its n_i neighbor atoms,

$$\mathscr{R}^i = \begin{pmatrix} \mathbf{R}_{1i} \\ \mathbf{R}_{2i} \\ \cdots \\ \mathbf{R}_{n_i i} \end{pmatrix} = \begin{pmatrix} x_{1i} & y_{1i} & z_{1i} \\ x_{2i} & y_{2i} & z_{2i} \\ \cdots & \cdots & \cdots \\ x_{n_i i} & y_{n_i i} & z_{n_i i} \end{pmatrix}. \tag{7}$$

Next, the local environment matrix $\mathscr{R}^i$ is transformed to the generalized local environment matrix $\tilde{\mathscr{R}}^i$,

$$\tilde{\mathscr{R}}^i = \begin{pmatrix} s(R_{1i}) & s(R_{1i})\frac{x_{1i}}{R_{1i}} & s(R_{1i})\frac{y_{1i}}{R_{1i}} & s(R_{1i})\frac{z_{1i}}{R_{1i}} \\ s(R_{2i}) & s(R_{2i})\frac{x_{2i}}{R_{2i}} & s(R_{2i})\frac{y_{2i}}{R_{2i}} & s(R_{2i})\frac{z_{2i}}{R_{2i}} \\ \cdots & \cdots & \cdots & \cdots \\ s(R_{n_i i}) & s(R_{n_i i})\frac{x_{n_i i}}{R_{n_i i}} & s(R_{n_i i})\frac{y_{n_i i}}{R_{n_i i}} & s(R_{n_i i})\frac{z_{n_i i}}{R_{n_i i}} \end{pmatrix}, \tag{8}$$

where $R_{ji} = \lVert \mathbf{R}_{ji} \rVert$ and

$$s(R_{ji}) = \begin{cases} \frac{1}{R_{ji}}, & R_{ji} < R_{cs}, \\ \frac{1}{R_{ji}}\{\frac{1}{2}\cos[\pi\frac{(R_{ji}-R_{cs})}{(R_c-R_{cs})}]+\frac{1}{2}\}, & R_{cs} < R_{ji} < R_c, \\ 0, & R_{ji} > R_c. \end{cases} \tag{9}$$

Here, R_{cs} is the switching distance from which the components in $\tilde{\mathscr{R}}^i$ smoothly decay to zero at the cut-off distance R_c. In this tutorial, we focused on relatively small molecular systems, and no cutoff was applied for the interatomic interactions, that is, $s(R_{ji}) = \frac{1}{R_{ji}}$ for any R_{ji} values.

In the next step of feature abstraction, an embedding neural network (ENN) G^{α_j,α_i} is used to map each $s(R_{ji})$ value through multiple hidden layers of neurons into m_1 outputs, which form the j-th row of the embedding matrix g_i. It should be noted that a separate embedding neural network G^{α_j,α_i} needs to be trained for each pair of the atom element types (α_j, α_i).

$$g_i = \begin{bmatrix} (G[s(R_{1i})])_1 & (G[s(R_{1i})])_2 & \dots & (G[s(R_{1i})])_{m_1} \\ (G[s(R_{2i})])_1 & (G[s(R_{2i})])_2 & \dots & (G[s(R_{2i})])_{m_1} \\ \dots & \dots & \dots & \dots \\ (G[s(R_{n_i i})])_1 & (G[s(R_{n_i i})])_2 & \dots & (G[s(R_{n_i i})])_{m_1} \end{bmatrix} \tag{10}$$

Lastly, a feature matrix D_i of size m_1 by m_2 is computed

$$D_i = (g_i^1)^{\mathrm{T}} \tilde{\mathscr{R}}^i (\tilde{\mathscr{R}}^i)^{\mathrm{T}} g_i^2, \tag{11}$$

where g_i^1 is the same as g_i (in Equation 10) and a submatrix g_i^2 contains the first m_2 columns of g_i (i.e., $m_2 \le m_1$). Both m_1 and m_2 are additional hyperparameters of the DeepPot-SE representation, besides the number of hidden layers and the number of neurons in each layer of the embedding networks.

2.2 | Feedforward neural networks

BPNN, named after Behler and Parrinello, was proposed in 2007 to deal with the difficulty in handling a varying number of atoms in molecules and permutation variance.[3,17,41] The basic idea of BPNN is to decompose the molecular energy (E) into atomic contributions (E_i)

$$E = \sum_{i=1}^{N} E_i, \tag{12}$$

where N is the total number of atoms in the molecule, and E_i is the energy of the ith atom as the output of a trained neural network. The input to the neural network is the atomic feature vector denoted as $\{\mathbf{G}_i\}$, like those defined in Equations (1) and (3), instead of the original molecular coordinates. The workflow of BPNN is illustrated in Figure 1, where an element-dependent feedforward neural network (S_i) maps the atomic feature vectors of the ith atom into its atomic energies (E_i).

The fitting networks for DeepPot-SE are similar to the neural networks in BPNN; the atomic feature vectors $\{\mathbf{G}_i\}$ are replaced with vectors that are reshaped from the feature matrix D_i for each atom in Equation (11).

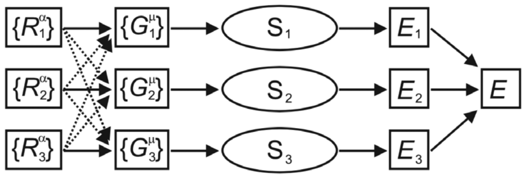

FIGURE 1 The Neural Network proposed by Behler and Parrinello (Figure 2 in Reference 17).

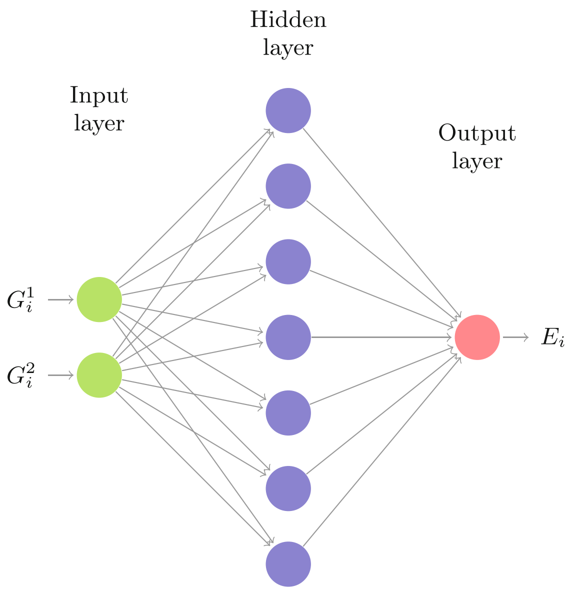

FIGURE 2 A fully-connected feedforward neural network with one hidden layer.

The standard structure of the neural network can be found in many books and articles.[3,42] A simple example of a NN with only one hidden layer is shown in Figure 2. With this NN, the molecular potential energy surface can be expressed as

$$E_i = \sum_{k=1}^{K} w_k f\left(\sum_{j=1}^{M} w_{jk} G_i^j + b_k\right) + b, \tag{13}$$

where K and M are the numbers of neurons in the hidden layer and input layer, respectively. G_i is concatenated feature vector of the i^{th} atom from Equations (1) and (3) (or 6), w_{jk} is the weight connecting node j in the input layer and node k in the hidden layer, and b_k is the bias of node k in the hidden layer. Similarly, w_k is the weight that connects node k in the hidden layer and the output layer (only one node), and b is the bias of the output layer. The activation function $f(x)$ can be an arbitrary nonlinear function and must be differentiable, such as a sigmoid function, a hyperbolic tangent function, a trigonometric function, or an exponential function. Nonlinearity ensures the complexity of NN, and the differentiability ensures that the parameters of a model can be optimized by the gradient descent method. The second derivatives of the activation functions should be available if the forces are used for the NN training.[43–45] The initial values of weight and bias parameters can be set randomly and are optimized during a training process using back-propagation.

The loss function is defined as the mean-squared error (MSE) of the predicted molecular energies with respect to those from reference quantum mechanical calculations as follows:

$$L = \frac{1}{N_s} \sum_{t=1}^{N_s} \left(E^t - E_{\text{ref}}^t\right)^2 \tag{14}$$

where N_s is the batch size, and E^t and E_{ref}^t are the potential energy predicted by the neural network and the reference electronic energy from a quantum mechanical calculation (ground truth) for the t^{th} structure, respectively. In the training of machine-learning potentials for driving molecular dynamics simulations, the loss function in Equation (14) is often augmented by the error in the predicted atomic forces.

2.3 | Gaussian process regression

Gaussian process regression (GPR) offers an alternative approach to modeling the relationship between molecular descriptors and the potential energy surface (PES).[46–55] The developments and applications of GPR for materials and molecules have recently been reviewed by Deringer et al.[40] Below, we will provide only a brief review of the GPR formalism that is relevant to this tutorial.

GPR is a non-parametric, kernel-based stochastic inference machine learning method.[39] Unlike NNs, which are optimized by minimizing the loss function that parameterizes a predefined functional form based on a predefined network architecture, GPR maximizes the likelihood of observations $\mathbf{y}$ (such as molecular energy) based on an infinite set of Gaussian-correlated latent functions. To begin, a prior distribution is assumed as follows:

$$\mathbf{f}(\mathbf{G}) \sim \mathcal{N}(\mathbf{0}, \mathbf{K}(\mathbf{G}, \mathbf{G})), \tag{15}$$

where $\mathbf{G}$ is a set of N_s d-dimensional input vectors $\mathbf{G} = [\mathbf{g}_1, \ldots, \mathbf{g}_{N_s}] = [g_{1,1}, \ldots, g_{1,d}, \ldots, g_{N_s,1}, \ldots, g_{N_s,d}]$, $\mathbf{0}$ is the mean of the functions and $\mathbf{K}$ is the covariance kernel matrix of the training data set based on a given covariance kernel function k that defines the similarity between the two input vectors involved:[39]

$$\mathbf{K}(\mathbf{G}, \mathbf{G}) = \begin{bmatrix} k(\mathbf{g}_1, \mathbf{g}_1) & \ldots & k(\mathbf{g}_1, \mathbf{g}_{N_s}) \\ \vdots & \ddots & \vdots \\ k(\mathbf{g}_{N_s}, \mathbf{g}_1) & \ldots & k(\mathbf{g}_{N_s}, \mathbf{g}_{N_s}) \end{bmatrix}. \tag{16}$$

In this tutorial, the covariance function, k, in use is the radial basis function:

$$k(\mathbf{g}_i, \mathbf{g}_j) = \sigma_f^2 \exp\left(-\frac{\|\mathbf{g}_i, \mathbf{g}_j\|^2}{2l^2}\right), \tag{17}$$

where σ_f^2 is the vertical variation parameter, l is the length scale parameter, and $||\mathbf{g}_i,\mathbf{g}_j||$ is the Euclidean distance between two input vectors $\mathbf{g}_i$ and $\mathbf{g}_j$. The third parameter, σ_n, is introduced to account for a certain level of noise in the observations, which modifies the covariance kernel matrix of the training data as follows:

$$\tilde{\mathbf{K}} = \mathbf{K}(\mathbf{G},\mathbf{G}) + \sigma_n^2\mathbf{I}, \tag{18}$$

where $\mathbf{I}$ is the identity matrix.[39] The hyperparameters, $\Theta = \{\sigma_f^2, l, \sigma_n^2\}$, are trained by maximizing the log marginal likelihood:

$$\log p(\mathbf{y}|\mathbf{G},\Theta) = -\frac{1}{2}\mathbf{y}^{\mathrm{T}}(\tilde{\mathbf{K}})^{-1}\mathbf{y} - \frac{1}{2}\log|\tilde{\mathbf{K}}| - \frac{N_s}{2}\log 2\pi, \tag{19}$$

The expected (E) energy at a new configuration $\mathbf{g}^*$ can be predicted by GPR as follows:

$$\mathrm{E}[f(\mathbf{g}^*)|\mathbf{y},\mathbf{g}^*,\mathbf{G},\Theta] = \mathbf{K}^*(\mathbf{K}+\sigma_n^2\mathbf{I})^{-1}\mathbf{y}, \tag{20}$$

where $\mathbf{K}^* = \mathbf{K}(\mathbf{g}^*,\mathbf{G})$. The variance of the predictive distribution can also be determined as follows:

$$\mathrm{Var}[f(\mathbf{g}^*)|\mathbf{y},\mathbf{g}^*,\mathbf{G},\Theta] = k(\mathbf{g}^*,\mathbf{g}^*) - \mathbf{K}^*(\mathbf{K}+\sigma_n^2\mathbf{I})^{-1}\mathbf{K}^{*\mathrm{T}}. \tag{21}$$

The forces are then calculated following the analytical gradient of the energy with respect to the Cartesian coordinates as follows:

$$F_{q_a} = -\sum_{j=1}^{d}\frac{\partial f(\mathbf{g}^*)}{\partial g_{*j}}\frac{\partial g_{*j}}{\partial q_a}. \tag{22}$$

Here, $f(\mathbf{g}^*)$ is the mean of the predictive distribution for energy, g_{*j} is the jth component of $\mathbf{g}^*$, and q_a corresponds to the Cartesian direction q (=x, y, or z) on atom a.

Similar to BPNN, permutational invariance can be introduced by adopting the Gaussian approximation potential (GAP) formalism developed by Bartók and co-workers.[47] This formalism utilizes a set of linear combination matrices, $\mathbf{L}$, to combine atomic contributions to the potential energy. Atomic contributions (ϵ) to the energy can be made according to

$$\epsilon(G^*) = \mathbf{k}^{*\mathrm{T}}\mathbf{L}(\mathbf{L}^{\mathrm{T}}\mathbf{K}_{nn}\mathbf{L}+\sigma_n^2\mathbf{I})^{-1}\mathbf{y}, \tag{23}$$

where G^* corresponds to the concatenated feature vector and $\mathbf{K}_{nn}$ is now a covariance matrix comparing each individual atomic environment.

In addition to being trained based on energy-only observations $\mathbf{y}$,[54] the GPR model can be influenced by including force observations in training, as demonstrated in our recent QM/MM work.[55] Because the derivatives of Gaussian processes, $\frac{\partial \mathbf{f}(\mathbf{G})}{\partial g}$, are also Gaussian processes, the observation set can be extended to include a set of derivative observations.[56] Here, we use M nuclear gradient components, $\frac{\partial f(\mathbf{g})}{\partial q_a}$, as our observable derivatives, and include them in an extended set, $\mathbf{y}_{\mathrm{ext}}$:

$$\mathbf{y}_{\mathrm{ext}} = \left[y_1,\ldots,y_{N_s},\frac{\partial f(\mathbf{g}_1)}{\partial q_1},\ldots,\frac{\partial f(\mathbf{g}_{N_s})}{\partial q_1},\ldots,\frac{\partial f(\mathbf{g}_1)}{\partial q_M},\ldots,\frac{\partial f(\mathbf{g}_{N_s})}{\partial q_M}\right]^{\mathrm{T}}. \tag{24}$$

The kernel is similarly extended, following the formalism introduced by Meyer and Hausser,[57] to account for the transformation between Cartesian ($\mathbf{Q}$) and internal ($\mathbf{G}$) input space:

$$\mathbf{K}_{\mathrm{ext}} = \begin{bmatrix} \mathbf{K}(\mathbf{G},\mathbf{G}') & \frac{\partial \mathbf{K}(\mathbf{G},\mathbf{G}')}{\partial \mathbf{Q}'} \\ \frac{\partial \mathbf{K}(\mathbf{G},\mathbf{G}')}{\partial \mathbf{Q}} & \frac{\partial^2 \mathbf{K}(\mathbf{G},\mathbf{G}')}{\partial \mathbf{Q}\partial \mathbf{Q}'} \end{bmatrix}. \tag{25}$$

After the model is optimized, the expected energy at a new configuration $\mathbf{g}^*$ can be predicted by GPR according to

$$\mathrm{E}[f(\mathbf{g}^*)|\mathbf{y}_{\mathrm{ext}},\mathbf{g}^*,\mathbf{G},\Theta] = \mathbf{K}_{\mathrm{ext}}^*(\mathbf{K}_{\mathrm{ext}}+\sigma_n^2\mathbf{I})^{-1}\mathbf{y}_{\mathrm{ext}}, \tag{26}$$

where $\mathbf{K}_{\mathrm{ext}}^* = \mathbf{K}_{\mathrm{ext}}(\mathbf{g}^*,\mathbf{G})$. The associated predictive variance is given by

$$\begin{aligned}&\mathrm{Var}[f(\mathbf{g}^*)|\mathbf{y}_{\mathrm{ext}},\mathbf{g}^*,\mathbf{G},\Theta] \\ &= k(\mathbf{g}^*,\mathbf{g}^*) - \mathbf{K}_{\mathrm{ext}}^*(\mathbf{K}_{\mathrm{ext}}+\sigma_n^2\mathbf{I})^{-1}\mathbf{K}_{\mathrm{ext}}^{*\mathrm{T}}.\end{aligned} \tag{27}$$

The prediction of the expected gradient is given by

$$\mathrm{E}\left[\frac{\partial f(\mathbf{g}^*)}{\partial q_a}|\mathbf{y}_{\mathrm{ext}},\mathbf{g}^*,\mathbf{G},\Theta\right] = \frac{\partial \mathbf{K}_{\mathrm{ext}}^*}{\partial q_a}(\mathbf{K}_{\mathrm{ext}}+\sigma_n^2\mathbf{I})^{-1}\mathbf{y}_{\mathrm{ext}}, \tag{28}$$

with the associated variance being

$$\begin{aligned}&\mathrm{Var}\left[\frac{\partial f(\mathbf{g}^*)}{\partial q_a}|\mathbf{y}_{\mathrm{ext}},\mathbf{g}^*,\mathbf{G},\Theta\right] \\ &= \frac{\partial^2 k(\mathbf{g}^*,\mathbf{g}^*)}{\partial q_a^2} - \frac{\partial \mathbf{K}_{\mathrm{ext}}^*}{\partial q_a}(\mathbf{K}_{\mathrm{ext}}+\sigma_n^2\mathbf{I})^{-1}\frac{\partial \mathbf{K}_{\mathrm{ext}}^{*\mathrm{T}}}{\partial q_a}.\end{aligned} \tag{29}$$

3 | LESSONS ON THE MLP TRAINING FOR MODEL SYSTEMS

The tutorials developed here are based on Jupyter notebooks and Google Colaboratory. Jupyter notebooks are a powerful and versatile tool popular in both research and education. They allow users to combine code, text, equations, and visualizations in a single interactive document, making them a great tool for exploring and understanding complex concepts. The interactive nature of Jupyter notebooks makes them particularly useful in education, as they allow students to experiment with code and see the results of their work in real time. Many universities and other educational institutions use Jupyter notebooks in their courses. Google Colaboratory, or Colab for short, is a popular hosted version of Jupyter notebooks that allows users to access powerful computing resources without having to set up and maintain their own infrastructure. Overall, Jupyter notebooks and Colab are valuable tools that can make learning more engaging and effective.

The tutorials will cover several important topics in machine learning and molecular modeling. In Lessons 1 and 2, we will introduce the concepts of neural networks and GPR and use these models to reproduce the two-dimensional Müller-Brown potential energy surface. In Lessons 3 and 4, we will introduce two molecular representations, the Behler-Parrinello symmetry functions and the Deep Potential, and explore their properties using the butane molecule as a test case. In the final three lessons (Lessons 5–7), we will combine these machine learning models and molecular representations to train several machine learning potentials that can accurately model the Claisen rearrangement reaction in the gas phase. Throughout the tutorials, we will provide hands-on examples that will allow students to apply what they have learned and gain practical experience with these important tools.

Lesson 1: Basic feedforward neural network models

We introduce the Müller-Brown potential energy surface and feedforward neural networks. A feedforward neural network is trained using grid points uniformly sampled from the Müller-Brown potential energy surface (Figure 3a). FNN models were trained using only energy (Figure 3b) and energy with gradient[58] (Figure 3c).

Lesson 2: Basic Gaussian process regression models

A GPR model is used to constructed a predicted surface (Figure 4) of the Müller-Brown potential energy surface, similar to Lesson 1. Emphasis is placed on GPR parameters, marginal likelihood, and variance from the analytical surface. Additional sections are added to show how gradients for a surface can be predicted using GPR.

Lesson 3: Behler-Parrinello symmetry functions for feature extraction

We introduce Behler-Parrinello[17] and ANI methods[18] for feature extraction using symmetry functions. This lesson discusses the importance of symmetry functions (Figure 5) for ensuring the energy of a molecule described by a neural network is rotationally and translationally invariant. BP and ANI are then used for feature extraction of a butane molecule. The parameters are set to values used in the ANI model.[18]

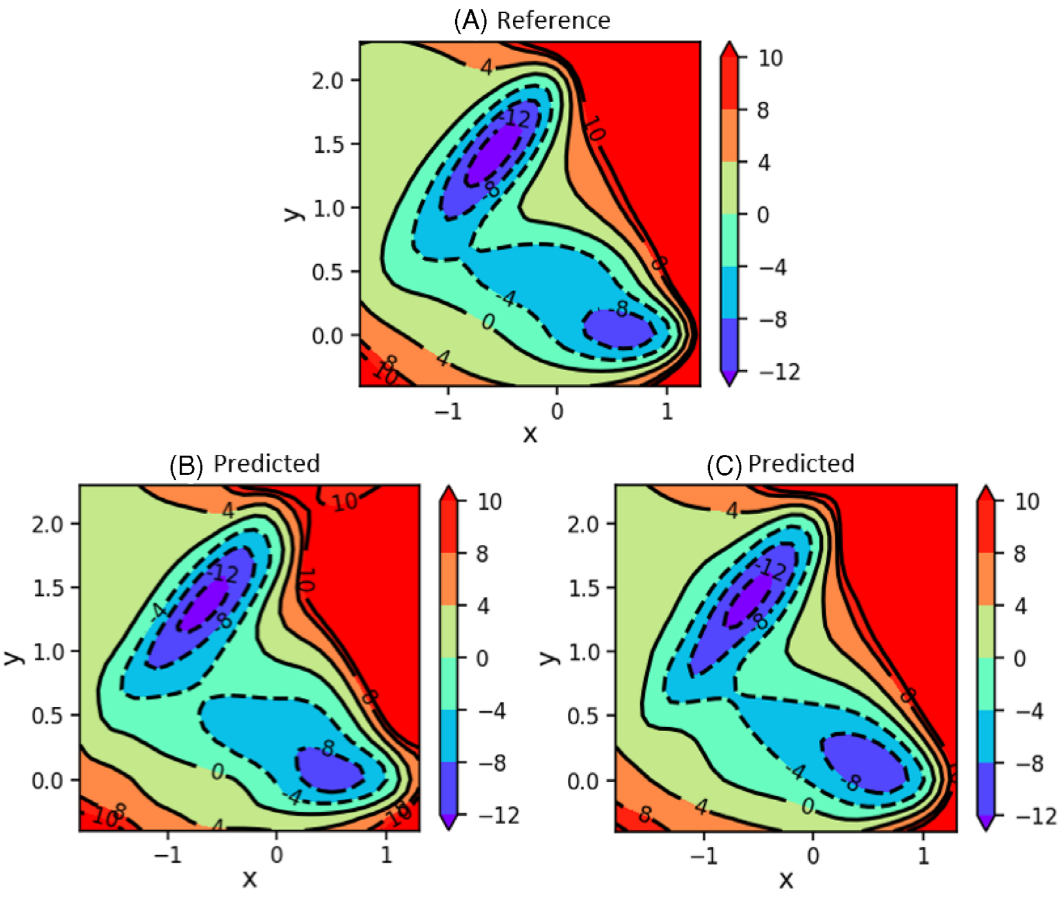

FIGURE 3 (A) Reference Müller-Brown PES, (B) predicted PES from the FNN trained with energies only, and (C) predicted PES from the FNN trained with energies and gradients.

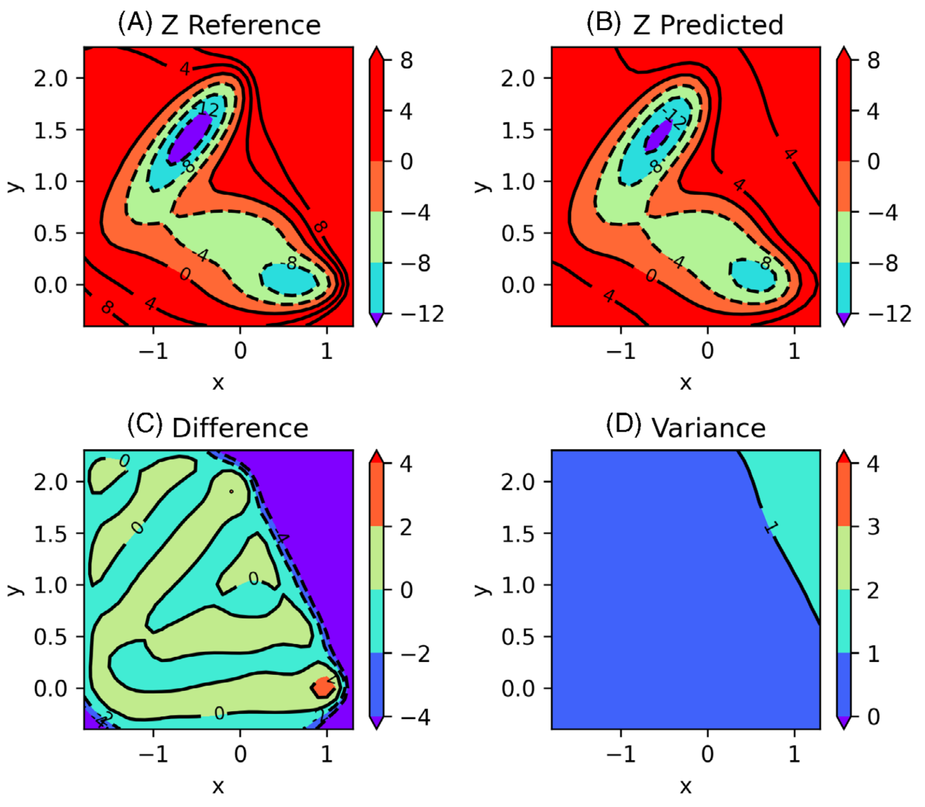

FIGURE 4 (A) Reference PES, (B) GPR predicted PES, (C) difference between the reference and GPR predicted surfaces, and (D) predicted variance for the Müller-Brown PES.

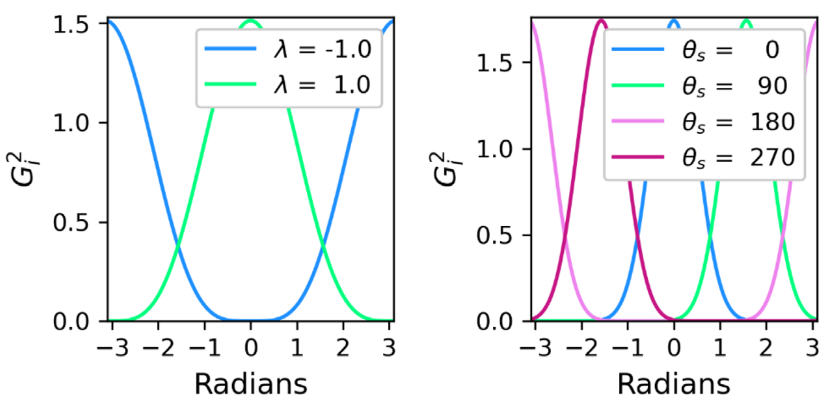

FIGURE 5 The BP angular symmetry functions (left) compared to the ANI angular symmetry functions (right).

Lesson 4: DeepPot representation for feature extraction

We provide an overview of DeepPot MLP training workflow (Figure 6) and discuss the significance of the embedding matrices for feature extraction in an embedding neural network. DeepPot is then used for feature extraction of butane molecular configurations from Lesson 3.

Lesson 5: BP-FNN models for the Claisen rearrangement

We combine the Behler-Parrinello symmetry functions with a feedforward neural network to construct a neural network that is rotationally and translationally invariant. The BP-FNN MLP is trained using geometries relevant to a Claisen rearrangement reaction (Figure 7). Following the training, the model is compared to reference values calculated using density functional theory (DFT) with B3LYP functional and 6-31+G* basis set.

Lesson 6: DeepPot-FNN models for the Claisen rearrangement

We combine DeepPot with a FNN to describe the molecular configurations along the Claisen rearrangement reaction pathway from Lesson 5. The predictions made by the DeepPot-FNN MLP model we train are again compared to DFT reference results.

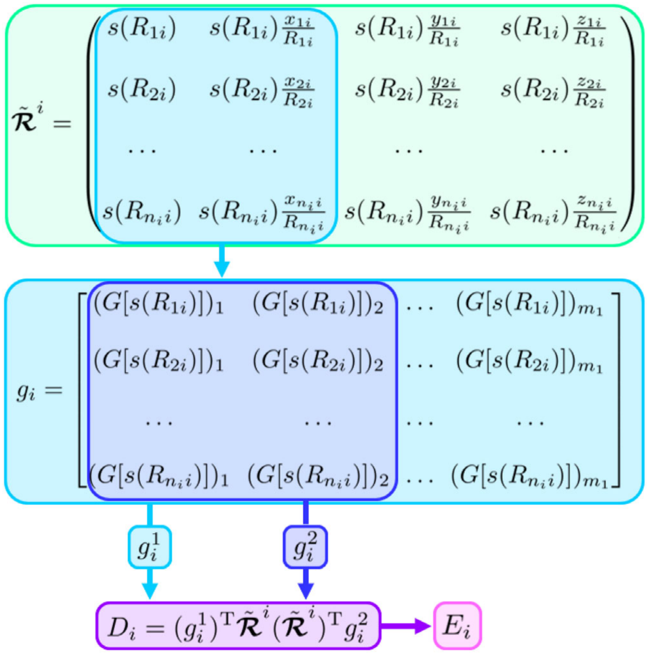
$$\tilde{\mathcal{R}}^i = \begin{pmatrix} s(R_{1i}) & s(R_{1i})\frac{x_{1i}}{R_{1i}} & s(R_{1i})\frac{y_{1i}}{R_{1i}} & s(R_{1i})\frac{z_{1i}}{R_{1i}} \\ s(R_{2i}) & s(R_{2i})\frac{x_{2i}}{R_{2i}} & s(R_{2i})\frac{y_{2i}}{R_{2i}} & s(R_{2i})\frac{z_{2i}}{R_{2i}} \\ \cdots & \cdots & \cdots & \cdots \\ s(R_{n_i i}) & s(R_{n_i i})\frac{x_{n_i i}}{R_{n_i i}} & s(R_{n_i i})\frac{y_{n_i i}}{R_{n_i i}} & s(R_{n_i i})\frac{z_{n_i i}}{R_{n_i i}} \end{pmatrix}$$
$$g_i = \begin{bmatrix} (G[s(R_{1i})])_1 & (G[s(R_{1i})])_2 & \cdots & (G[s(R_{1i})])_{m_1} \\ (G[s(R_{2i})])_1 & (G[s(R_{2i})])_2 & \cdots & (G[s(R_{2i})])_{m_1} \\ \cdots & \cdots & \cdots & \cdots \\ (G[s(R_{n_i i})])_1 & (G[s(R_{n_i i})])_2 & \cdots & (G[s(R_{n_i i})])_{m_1} \end{bmatrix}$$
g_i^1
g_i^2
$D_i = (g_i^1)^T\tilde{\mathcal{R}}^i(\tilde{\mathcal{R}}^i)^T g_i^2$
E_i

FIGURE 6 Schematic of the DeepPot-SE feature extraction process for the i^{th} atom.

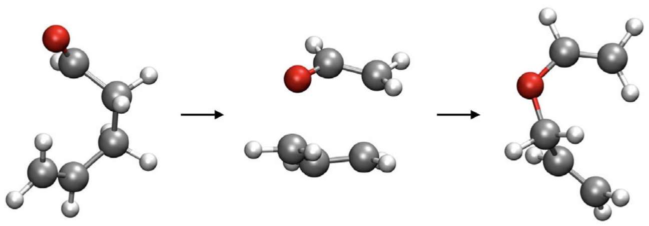

FIGURE 7 Claisen rearrangement modeled in lessons 5-7.

Lesson 7: BP-GPR models for the Claisen rearrangement

Our final lesson combines the BP and ANI symmetry functions with GPR to create an MLP model. The BP-GPR MLP model is trained and tested using the same reactive system as in Lessons 5 and 6.

4 | DISCUSSION

This tutorial focuses on the training of MLPs for describing the ground-state potential energy surface of a reactive system. It should be noted that our focus is placed on the readability of the code implementation, rather than the software modularity or run-time efficiency. Once learning the basics through this tutorial, the readers can adopt advanced software platforms, such as DeePMD-kit,[22,59,60] ænet,[61,62] AMP,[63] MLatom, PhysNet,[37] SchNetPack,[36] sGDML,[64] TorchANI,[20] and TorchMD-NET[65] for their own machine learning model development. It should also be noted that there are several other areas of research that are not covered. These include:

1. MLPs for describing electronic excited states. A comprehensive review on this topic can be found in Reference 11. In general, it would require the training of several MLPs, one for each adiabatic or diabatic electronic surface, as well as, in the former case, the training of ML models for the non-adiabatic coupling.[66–72]
2. Active learning/adaptive sampling schemes for training the MLPs for molecular dynamics simulations. This can involve (a) the estimation of prediction uncertainty using the query-of-the-committee[19,69,71,73] and other approaches[74] and (b) the use of uncertainty estimates in hyperactive learning to bias sampling toward large uncertainty regions in the generation of a training set.[75,76]
3. Efficient protocols for generating MLPs for QM/MM simulations. It is not practical to incorporate all MM atoms (in addition to QM atoms) in the training of these potentials, as this would lead to an explosively large array of descriptors, the most straightforward way is to include only MM atoms within a distance cutoff from the QM region in the MLP training.[77–79] In general, a smooth distance cutoff is necessary to ensure a smooth potential energy surface.[77,80,81] Alternatively, one can adopt an implicit description of the MM environment through the use of MM-perturbed semi-empirical QM charges,[82,83] MM electrostatic potential or field at QM atom positions,[84,85] or through polarizable embedding.[86] One can also use both MM electrostatic potential and field in the training of QM/MM MLPs[81,87] using our QM/MM-AC scheme[80] for separating inner and outer MM atoms and projecting outer MM charges onto inner MM atom positions.[80,88,89]

These topics will be covered in the future advanced tutorials on MLPs.

5 | CONCLUSIONS

A Colab tutorial was developed to showcase the implementation of basic machine learning models (neural networks; Gaussian process regression) for reactive systems (using Claisen arrangement as a model system). We hope this tutorial will make it easier for undergraduate/graduate students to get familiar with the basics of machine learning techniques in the context of atomistic modeling.

ACKNOWLEDGMENTS

We would like to dedicate this tutorial to Prof. Elfi Kraka on the occasion of her 70th birthday. Her dedication to theoretical and computational chemistry research and education has inspired us for years. We acknowledge the support from the National Institutes of Health through grants R01GM135392 (Shao and Pu), R44GM133270, and P30GM145423 (Shao). We also acknowledge the support from the National Science Foundation through grant CHE-2102071 (Shao) and the National Natural Science Foundation of China through grant

22073030 (Mei). Mao acknowledges the support from San Diego State University Startup Fund. Shao acknowledges the support from the OU Data Institute for the Societal Challenges monthly seed funding program. The authors thank the OU Supercomputing Center for Education & Research (OSCER) for the computational resources. Pu acknowledges the Indiana University Pervasive Technology Institute (supported in part by Lilly Endowment, Inc.) for providing supercomputing resources on BigRed3 and Carbonate that have facilitated the research reported in this paper.

DATA AVAILABILITY STATEMENT

The data that support the findings of this study are available at https://github.com/cc-ats/mlp_tutorial.

ORCID

Xiaoliang Pan https://orcid.org/0000-0002-6399-4853
Yuezhi Mao https://orcid.org/0000-0001-5362-4333
Ye Mei https://orcid.org/0000-0002-3953-8508
Jingzhi Pu https://orcid.org/0000-0002-3042-335X
Yihan Shao https://orcid.org/0000-0001-9337-341X

Bibliography

[1] J. Behler, *Phys. Chem. Chem. Phys.* **2011**, *13*, 17930.
[2] S. Manzhos, R. Dawes, T. Carrington, *Int. J. Quantum Chem.* **2015**, *115*, 1012.
[3] J. Behler, *Int. J. Quantum Chem.* **2015**, *115*, 1032.
[4] J. Behler, *J. Chem. Phys.* **2016**, *145*, 170901.
[5] B. Jiang, J. Li, H. Guo, *Int. Rev. Phys. Chem.* **2016**, *35*, 479.
[6] K. T. Butler, D. W. Davies, H. Cartwright, O. Isayev, A. Walsh, *Nature* **2018**, *559*, 547.
[7] P. O. Dral, *J. Phys. Chem. Lett.* **2020**, *11*, 2336.
[8] J. Zhang, Y.-K. Lei, Z. Zhang, J. Chang, M. Li, X. Han, L. Yang, Y. I. Yang, Y. Q. Gao, *J. Phys. Chem. A* **2020**, *124*, 6745.
[9] M. Pinheiro, F. Ge, N. Ferré, P. O. Dral, M. Barbatti, *Chem. Sci.* **2021**, *12*, 14396.
[10] J. Westermayr, M. Gastegger, K. T. Schütt, R. J. Maurer, *J. Chem. Phys.* **2021**, *154*, 230903.
[11] J. Westermayr, P. Marquetand, *Chem. Rev.* **2021**, *121*, 9873.
[12] H. J. Kulik, T. Hammerschmidt, J. Schmidt, S. Botti, M. A. L. Marques, M. Boley, M. Scheffler, M. Todorović, P. Rinke, C. Oses, A. Smolyanyuk, S. Curtarolo, A. Tkatchenko, A. P. Bartók, S. Manzhos, M. Ihara, T. Carrington, J. Behler, O. Isayev, M. Veit, A. Grisafi, J. Nigam, M. Ceriotti, K. T. Schütt, J. Westermayr, M. Gastegger, R. J. Maurer, B. Kalita, K. Burke, R. Nagai, R. Akashi, O. Sugino, J. Hermann, F. Noé, S. Pilati, C. Draxl, M. Kuban, S. Rigamonti, M. Scheidgen, M. Esters, D. Hicks, C. Toher, P. V. Balachandran, I. Tamblyn, S. Whitelam, C. Bellinger, L. M. Ghiringhelli, *Electron. Struct.* **2022**, *4*, 23004.
[13] H. Gokcan, O. Isayev, *WIREs Comput. Mol. Sci.* **2022**, *12*, e1564.
[14] J. M. Bowman, C. Qu, R. Conte, A. Nandi, P. L. Houston, Q. Yu, *J. Chem. Theory Comput.* **2023**, *19*, 1.
[15] R. Biswas, U. Lourderaj, N. Sathyamurthy, *J. Chem. Sci.* **2023**, *135*, 22.
[16] H. Bhatia, F. Aydin, T. S. Carpenter, F. C. Lightstone, P.-T. Bremer, H. I. Ingólfsson, D. V. Nissley, F. H. Streitz, *Curr. Opin. Struct. Biol.* **2023**, *80*, 102569.
[17] J. Behler, M. Parrinello, *Phys. Rev. Lett.* **2007**, *98*, 146401.
[18] J. S. Smith, O. Isayev, A. E. Roitberg, *Chem. Sci.* **2017**, *8*, 3192.
[19] J. S. Smith, B. T. Nebgen, R. Zubatyuk, N. Lubbers, C. Devereux, K. Barros, S. Tretiak, O. Isayev, A. E. Roitberg, *Nat. Commun.* **2019**, *10*, 2903.
[20] X. Gao, F. Ramezanghorbani, O. Isayev, J. S. Smith, A. E. Roitberg, *J. Chem. Inf. Model.* **2020**, *60*, 3408.
[21] L. Zhang, J. Han, H. Wang, W. A. Saidi, R. Car, E. Weinan, *NeurIPS* **2018**, *31*, 4441.
[22] H. Wang, L. Zhang, J. Han, E. Weinan, *Comput. Phys. Commun.* **2018**, *228*, 178.
[23] Y. Zhang, H. Wang, W. Chen, J. Zeng, L. Zhang, H. Wang, E. Weinan, *Comput. Phys. Commun.* **2020**, *253*, 107206.
[24] J. Zeng, Y. Tao, T. J. Giese, D. M. York, *J. Chem. Theory Comput.* **2023**, *19*, 1261.
[25] J. Zeng, Y. Tao, T. J. Giese, D. M. York, *J. Chem. Phys.* **2023**, *158*, 124110.
[26] S. Manzhos, X. Wang, R. Dawes, T. Carrington, *J. Phys. Chem. A* **2006**, *110*, 5295.
[27] M. Rupp, A. Tkatchenko, K.-R. Müller, O. A. von Lilienfeld, *Phys. Rev. Lett.* **2012**, *108*, 58301.
[28] B. Jiang, H. Guo, *J. Chem. Phys.* **2013**, *139*, 054112.
[29] C. Qu, P. L. Houston, R. Conte, A. Nandi, J. M. Bowman, *J. Phys. Chem. Lett.* **2021**, *12*, 4902.
[30] K. Hansen, F. Biegler, R. Ramakrishnan, W. Pronobis, O. A. von Lilienfeld, K.-R. Müller, A. Tkatchenko, *J. Phys. Chem. Lett.* **2015**, *6*, 2326.
[31] P. O. Dral, A. Owens, S. N. Yurchenko, W. Thiel, *J. Chem. Phys.* **2017**, *146*, 244108.
[32] F. A. Faber, A. S. Christensen, B. Huang, O. A. von Lilienfeld, *J. Chem. Phys.* **2018**, *148*, 241717.
[33] M. Gastegger, L. Schwiedrzik, M. Bittermann, F. Berzsenyi, P. Marquetand, *J. Chem. Phys.* **2018**, *148*, 241709.
[34] K. T. Schütt, F. Arbabzadah, S. Chmiela, K. R. Müller, A. Tkatchenko, *Nat. Commun.* **2017**, *8*, 13890.
[35] K. T. Schütt, H. E. Sauceda, P.-J. Kindermans, A. Tkatchenko, K.-R. Müller, *J. Chem. Phys.* **2018**, *148*, 241722.
[36] K. T. Schütt, P. Kessel, M. Gastegger, K. A. Nicoli, A. Tkatchenko, K.-R. Müller, *J. Chem. Theory Comput.* **2019**, *15*, 448.
[37] O. T. Unke, M. Meuwly, *J. Chem. Theory Comput.* **2019**, *15*, 3678.
[38] S. Batzner, A. Musaelian, L. Sun, M. Geiger, J. P. Mailoa, M. Kornbluth, N. Molinari, T. E. Smidt, B. Kozinsky, *Nat. Commun.* **2022**, *13*, 2453.
[39] C. E. Rasmussen, C. K. Williams, *Gaussian Processes for Machine Learning*, MIT Press, Cambridge **2006**.
[40] V. L. Deringer, A. P. Bartok, N. Bernstein, D. M. Wilkins, M. Ceriotti, G. Csanyi, *Chem. Rev.* **2021**, *121*, 10073.
[41] J. Behler, *J. Chem. Phys.* **2011**, *134*, 074106.
[42] T. Hastie, R. Tibshirani, J. Friedman, *The Elements of Statistical Learning:Data Mining, Inference, and Prediction*, 2nd ed., Springer Science, New York, NY **2009**.
[43] N. Artrith, J. Behler, *Phys. Rev. B* **2012**, *85*, 045439.
[44] J. B. Witkoskie, D. J. Doren, *J. Chem. Theory Comput.* **2005**, *1*, 14.
[45] A. Pukrittayakamee, M. Malshe, M. Hagan, L. M. Raff, R. Narulkar, S. Bukkapatnam, R. Komanduri, *J. Chem. Phys.* **2009**, *130*, 134101.
[46] B. Kolb, P. Marshall, B. Zhao, B. Jiang, H. Guo, *J. Phys. Chem. A* **2017**, *121*, 2552.
[47] A. P. Bartók, M. C. Payne, R. Kondor, G. Csányi, *Phys. Rev. Lett.* **2010**, *104*, 136403.
[48] S. De, A. P. Bartók, G. Csányi, M. Ceriotti, *Phys. Chem. Chem. Phys.* **2016**, *18*, 13754.
[49] E. Uteva, R. S. Graham, R. D. Wilkinson, R. J. Wheatley, *J. Chem. Phys.* **2018**, *149*, 174114.
[50] K. Rossi, V. Jurásková, R. Wischert, L. Garel, C. Corminbœuf, M. Ceriotti, *J. Chem. Theory Comput.* **2020**, *16*, 5139.
[51] A. S. Christensen, L. A. Bratholm, F. A. Faber, O. Anatole von Lilienfeld, *J. Chem. Phys.* **2020**, *152*, 44107.
[52] J. Vandermause, S. B. Torrisi, S. Batzner, Y. Xie, L. Sun, A. M. Kolpak, B. Kozinsky, *Npj Comput. Mater.* **2020**, *6*, 20.
[53] B. C. B. Symons, M. K. Bane, P. L. A. Popelier, *J. Chem. Theory Comput.* **2021**, *17*, 7043.

[54] R. Snyder, B. Kim, X. Pan, Y. Shao, J. Pu, *J. Chem. Phys.* **2023**, *159*, 54107.
[55] R. Snyder, B. Kim, X. Pan, Y. Shao, J. Pu, *Phys. Chem. Chem. Phys.* **2022**, *24*, 25134.
[56] E. Solak, R. Murray-Smith, W. E. Leithead, D. J. Leith, C. E. Rasmussen, *NIPS* **2003**, *15*, 1033.
[57] R. Meyer, A. W. Hauser, *J. Chem. Phys.* **2020**, *152*, 84112.
[58] J. S. Smith, N. Lubbers, A. P. Thompson, K. Barros, Simple and efficient algorithms for training machine learning potentials to force data. **2020** 10.48550/ARXIV.2006.05475, publisher: arXiv Version Number 1.
[59] W. Liang, J. Zeng, D. M. York, L. Zhang, H. Wang, in *A Practical Guide to Recent Advances in Multiscale Modeling and Simulation of Biomolecules* (Eds: Y. Wang, R. Zhou), (AIP Publishing LLCMelville, New York **2023** 6-1–6-20.
[60] J. Zeng, D. Zhang, D. Lu, P. Mo, Z. Li, Y. Chen, M. Rynik, L. Huang, Z. Li, S. Shi, Y. Wang, H. Ye, P. Tuo, J. Yang, Y. Ding, Y. Li, D. Tisi, Q. Zeng, H. Bao, Y. Xia, J. Huang, K. Muraoka, Y. Wang, J. Chang, F. Yuan, S. L. Bore, C. Cai, Y. Lin, B. Wang, J. Xu, J.-X. Zhu, C. Luo, Y. Zhang, R. E. A. Goodall, W. Liang, A. K. Singh, S. Yao, J. Zhang, R. Wentzcovitch, J. Han, J. Liu, W. Jia, D. M. York, E. Weinan, R. Car, L. Zhang, H. Wang, *J. Chem. Phys.* **2023**, *159*, 54801.
[61] N. Artrith, A. Urban, *Comput. Mat. Sci.* **2016**, *114*, 135.
[62] J. López-Zorrilla, X. M. Aretxabaleta, I. W. Yeu, I. Etxebarria, H. Manzano, N. Artrith, *J. Chem. Phys.* **2023**, *158*, 164105.
[63] A. Khorshidi, A. A. Peterson, *Comput. Phys. Commun.* **2016**, *207*, 310.
[64] S. Chmiela, H. E. Sauceda, I. Poltavsky, K.-R. Müller, A. Tkatchenko, *Comput. Phys. Commun.* **2019**, *240*, 38.
[65] S. Doerr, M. Majewsk, A. Pérez, A. Krämer, C. Clementi, F. Noe, T. Giorgino, G. D. Fabritiis, Torchmd: A deep learning framework for molecular simulations. **2020** arXiv:2012.12106 [physics.chem-ph].
[66] W.-K. Chen, X.-Y. Liu, W.-H. Fang, P. O. Dral, G. Cui, *J. Phys. Chem. Lett.* **2018**, *9*, 6702.
[67] D. Hu, Y. Xie, X. Li, L. Li, Z. Lan, *J. Phys. Chem. Lett.* **2018**, *9*, 2725.
[68] P. O. Dral, M. Barbatti, W. Thiel, *J. Phys. Chem. Lett.* **2018**, *9*, 5660.
[69] J. Westermayr, M. Gastegger, M. F. S. J. Menger, S. Mai, L. González, P. Marquetand, *Chem. Sci.* **2019**, *10*, 8100.
[70] Y. Shen, D. R. Yarkony, *J. Phys. Chem. A* **2020**, *124*, 4539.
[71] J. Li, P. Reiser, B. R. Boswell, A. Eberhard, N. Z. Burns, P. Friederich, S. A. Lopez, *Chem. Sci.* **2021**, *12*, 5302.
[72] J.-K. Ha, K. Kim, S. K. Min, *J. Chem. Theory Comput.* **2021**, *17*, 694.
[73] H. S. Seung, M. Opper, H. Sompolinsky, *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, ACM, Pittsburgh, PA **1992**, p. 287.
[74] A. R. Tan, S. Urata, S. Goldman, J. C. B. Dietschreit, Gómez-Bombarelli, Single-model uncertainty quantification in neural network potentials does not consistently outperform model ensembles. **2023** 10.48550/ARXIV.2305.01754, publisher: arXiv Version Number: 1.
[75] C. van der Oord, M. Sachs, D. P. Kovács, C. Ortner, G. Csányi, Hyperactive learning (HAL) for data-driven interatomic potentials. **2022** arxiv: 10.48550/ARXIV.2210.0422, version number 2.
[76] M. Kulichenko, K. Barros, N. Lubbers, Y. W. Li, R. Messerly, S. Tretiak, J. S. Smith, B. Nebgen, *Nat. Comput. Sci.* **2023**, *3*, 230.
[77] J. Zeng, T. J. Giese, S. Ekesan, D. M. York, *J. Chem. Theory Comput.* **2021**, *17*, 6993.
[78] L. Böselt, M. Thürlemann, S. Riniker, *J. Chem. Theory Comput.* **2021**, *17*, 2641.
[79] B. Lier, P. Poliak, P. Marquetand, J. Westermayr, C. Oostenbrink, *J. Phys. Chem. Lett.* **2022**, *13*, 3812.
[80] X. Pan, K. Nam, E. Epifanovsky, A. C. Simmonett, E. Rosta, Y. Shao, *J. Chem. Phys.* **2021**, *154*, 24115.
[81] X. Pan, J. Yang, R. Van, E. Epifanovsky, J. Ho, J. Huang, J. Pu, Y. Mei, K. Nam, Y. Shao, *J. Chem. Theory Comput.* **2021**, *17*, 5745.
[82] L. Shen, J. Wu, W. Yang, *J. Chem. Theory Comput.* **2016**, *12*, 4934.
[83] J. Wu, L. Shen, W. Yang, *J. Chem. Phys.* **2017**, *147*, 161732.
[84] L. Shen, W. Yang, *J. Chem. Theory Comput.* **2018**, *14*, 1442.
[85] M. Gastegger, K. T. Schütt, K.-R. Müller, *Chem. Sci.* **2021**, *12*, 11473.
[86] K. Zinovjev, *J. Chem. Theory Comput.* **2023**, *19*, 1888.
[87] S. Yao, R. Van, X. Pan, J. H. Park, Y. Mao, J. Pu, Y. Mei, Y. Shao, *RSC Adv.* **2023**, *13*, 4565.
[88] B. A. Gregersen, D. M. York, *J. Phys. Chem. B* **2005**, *109*, 536.
[89] B. A. Gregersen, D. M. York, *J. Comput. Chem.* **2006**, *27*, 103.

How to cite this article: X. Pan, R. Snyder, J.-N. Wang, C. Lander, C. Wickizer, R. Van, A. Chesney, Y. Xue, Y. Mao, Y. Mei, J. Pu, Y. Shao, *J. Comput. Chem.* **2024**, *45*(10), 638. https://doi.org/10.1002/jcc.27269