

## Machine Learning Force Fields

Oliver T. Unke,<sup>△</sup> Stefan Chmiela,<sup>△</sup> Huziel E. Sauceda, Michael Gastegger, Igor Poltavsky, Kristof T. Schütt, Alexandre Tkatchenko,\* and Klaus-Robert Müller\*

Cite This: *Chem. Rev.* 2021, 121, 10142–10186

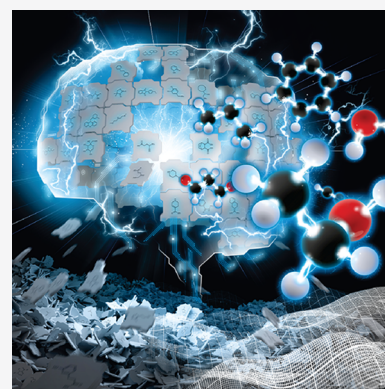
Read Online

ACCESS |

Metrics & More

Article Recommendations

**ABSTRACT:** In recent years, the use of machine learning (ML) in computational chemistry has enabled numerous advances previously out of reach due to the computational complexity of traditional electronic-structure methods. One of the most promising applications is the construction of ML-based force fields (FFs), with the aim to narrow the gap between the accuracy of *ab initio* methods and the efficiency of classical FFs. The key idea is to learn the statistical relation between chemical structure and potential energy without relying on a preconceived notion of fixed chemical bonds or knowledge about the relevant interactions. Such universal ML approximations are in principle only limited by the quality and quantity of the reference data used to train them. This review gives an overview of applications of ML-FFs and the chemical insights that can be obtained from them. The core concepts underlying ML-FFs are described in detail, and a step-by-step guide for constructing and testing them from scratch is given. The text concludes with a discussion of the challenges that remain to be overcome by the next generation of ML-FFs.



### CONTENTS

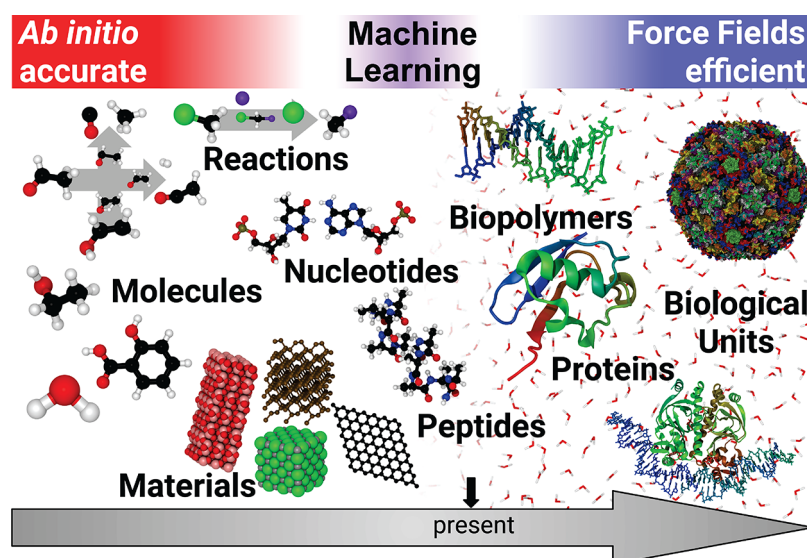
1. Introduction	10143	3.3.5. Normal Mode Sampling	10161
2. Mathematical and Conceptual Framework	10145	3.3.6. Problems Due to Insufficient Sampling	10161
2.1. Chemistry Foundations	10145	3.3.7. Importance of Data Consistency	10162
2.1.1. Potential Energy Surface	10145	3.4. Data Preparation	10162
2.1.2. Invariances of Physical Systems	10146	3.4.1. Splitting the Data	10162
2.2. Machine Learning Foundations	10146	3.4.2. Data Preprocessing	10163
2.2.1. Kernel-Based Methods	10147	3.5. Training the Model	10163
2.2.2. Artificial Neural Networks	10150	3.5.1. Choosing the Loss Function	10163
2.2.3. Model Selection: How to Choose Hyper-parameters	10150	3.5.2. Tuning Hyperparameters	10164
2.3. Combining Machine Learning and Chemistry	10152	3.5.3. Regularization	10164
2.3.1. Physical Constraints	10152	3.6. Using ML-FFs in Production	10165
2.3.2. (Symmetric) Gradient Domain Machine Learning ((s)GDML)	10154	3.7. Example Code and Software Packages	10165
2.3.3. Gaussian Approximation Potentials (GAPs)	10154	3.7.1. The sGDML Package	10165
2.3.4. Neural Network Potentials	10155	3.7.2. The SchNetPack Package	10166
3. Best Practices and Pitfalls	10157	3.7.3. Other Software Packages	10167
3.1. Preliminary Considerations	10157	4. Physical and Chemical Insights from Machine Learned Force Fields	10168
3.1.1. Practicability	10157	4.1. Electronic Effects	10169
3.1.2. Multireference Effects	10157	4.2. Thermodynamics	10169
3.1.3. Strong Delocalization	10157	4.3. Reactions	10170
3.2. Choosing an Appropriate ML Method	10157	4.4. Nuclear Quantum Effects	10171
3.3. Data Collection	10160	4.5. Excited States	10172
3.3.1. AIMD Sampling	10160		
3.3.2. Sampling by Proxy	10160		
3.3.3. Adaptive Sampling	10161		
3.3.4. Metadynamics Sampling	10161		

**Special Issue:** Machine Learning at the Atomic Scale

**Received:** October 12, 2020

**Published:** March 11, 2021





**Figure 1.** Accurate *ab initio* methods are computationally demanding and can only be used to study small systems in gas phase or regular periodic materials. Larger molecules in solution, such as proteins, are typically modeled by force fields, empirical functions that trade accuracy for computational efficiency. Machine learning methods are closing this gap and allow to study increasingly large chemical systems at *ab initio* accuracy with force field efficiency.

4.6. Spectroscopy	10172
5. Challenges	10173
5.1. Locality and Smoothness Assumptions	10173
5.2. Transferability, Scalability, and Long-Range Interactions	10175
6. Concluding Remarks	10177
Author Information	10178
Corresponding Authors	10178
Authors	10178
Author Contributions	10178
Notes	10178
Biographies	10178
Acknowledgments	10179
References	10179

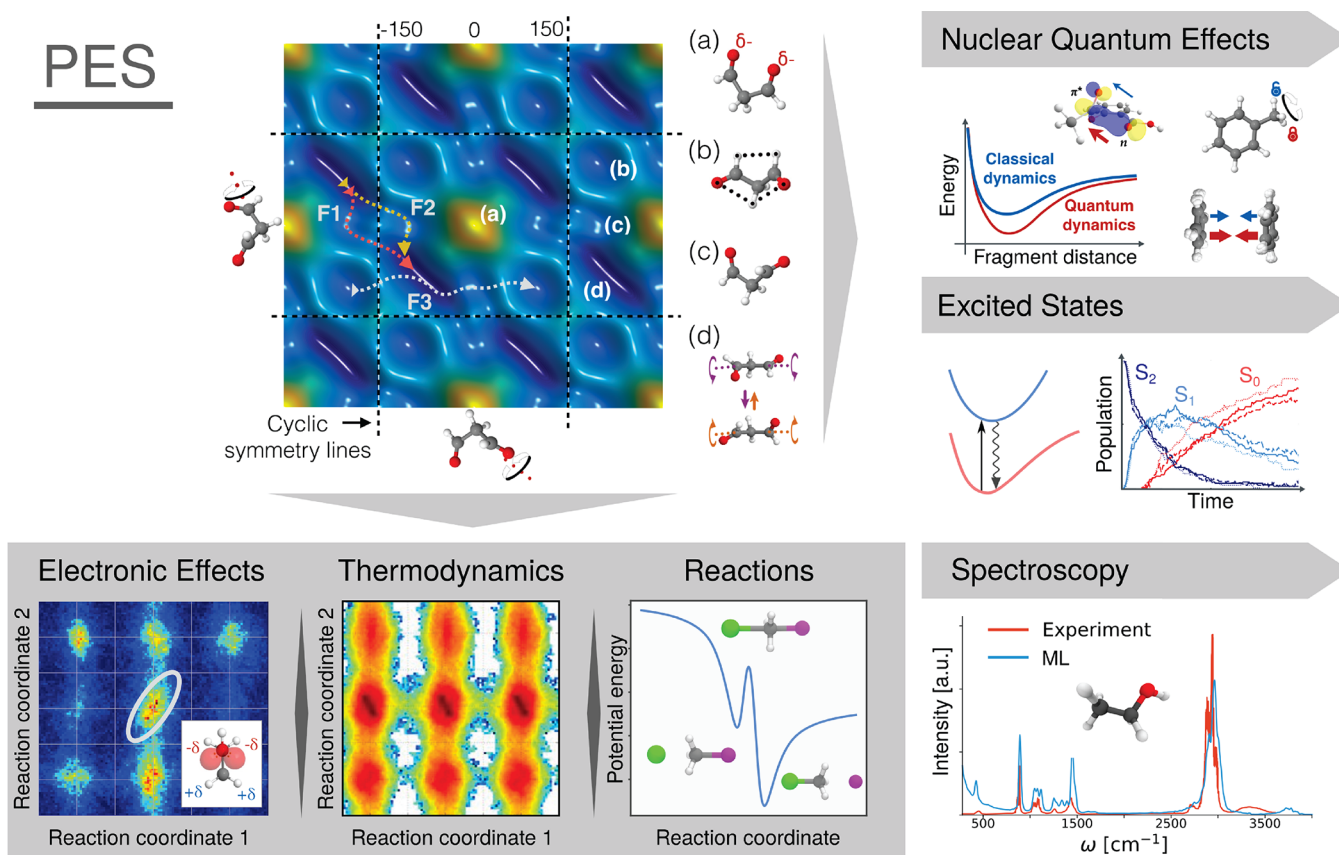
## 1. INTRODUCTION

In 1964, physicist Richard Feynman famously remarked “that all things are made of atoms, and that everything that living things do can be understood in terms of the jiggings and wiggings of atoms”.<sup>1</sup> As such, an atomically resolved picture can provide invaluable insights on biological (and other) processes. The first molecular dynamics (MD) study of a protein in 1977 by McCammon et al.<sup>2</sup> did not consider explicit solvent molecules and was limited to less than 10 ps of simulation. Still, it challenged the (at that time) common belief that proteins are essentially rigid structures<sup>3</sup> and, instead, suggested that the interior of proteins behaves more fluid-like. Since then, systems consisting of more than a million atoms have been studied,<sup>4</sup> simulation times extended to the millisecond regime,<sup>5</sup> and the study of entire viruses in atomic detail made possible.<sup>6,7</sup> Recently, a distributed computing effort even allowed to study the viral proteome of SARS-CoV-2 for a total of 0.1 s of simulation time.<sup>8</sup>

To perform MD simulations, typically, the Newtonian equations of motion are integrated numerically, which requires knowledge of the forces acting on individual atoms at each time step of the simulation.<sup>9</sup> In principle, the most accurate

way to obtain these forces is by solving the Schrödinger equation (SE), which describes the physical laws underlying most chemical phenomena and processes.<sup>10</sup> Unfortunately, an analytic solution is only possible for two-body systems such as the hydrogen atom. For larger chemical structures, the SE can only be solved approximately. However, even with approximations, an accurate numerical solution is a computationally demanding task. For example, the CCSD(T) method (coupled cluster with singles, doubles and perturbative triples), which is widely regarded as the “gold standard” of chemistry (as its predictions compare well with experimental results),<sup>11</sup> scales  $\propto N^7$  with the number of atoms  $N$ . (Strictly speaking, the true scaling of the CCSD(T) method is  $O(n^7)$ , where  $n$  is the number of basis functions used for the wave function ansatz. Depending on the desired accuracy and which atoms are present (more precisely, how many electrons are in their shells),  $n$  can vary greatly. However, the number of atoms is usually a good proxy.) Because of this, it is unfeasible to calculate the forces for many different configurations of large chemical systems, which is required for running MD simulations, with accurate methods. Instead, simple empirical functions are commonly used to model the relevant interactions. From these so-called force fields (FFs), atomic forces can be readily derived analytically.

Most conventional FFs model chemical interactions as a sum over bonded and nonbonded terms.<sup>12,13</sup> The former can be described with simple functions of the distances between directly bonded atoms, or angles and dihedrals between atoms sharing some of their bonding partners. The nonbonded terms consider pairwise combinations of atoms, typically by modeling electrostatics with Coulomb’s law (assuming a point charge at each atom’s position) and dispersion with a Lennard-Jones potential.<sup>14</sup> Because of the computational efficiency of these terms, such classical FFs allow to study systems consisting of many thousands of atoms. However, while offering a qualitatively reasonable description of chemical interactions, the quality of MD simulations, and hence the insights that can be obtained from them, are ultimately limited



**Figure 2.** ML-FFs combine the accuracy of *ab initio* methods and the efficiency of classical FFs. They provide easy access to a system's potential energy surface (PES), which can in turn be used to derive a plethora of other quantities. By using them to run MD simulations on a single PES, ML-FFs allow chemical insights inaccessible to other methods (see gray box). For example, they accurately model electronic effects and their influence on thermodynamic observables and allow a natural description of chemical reactions, which is difficult or even impossible with conventional FFs. Their efficiency also allows them to be applied in situations where the Born–Oppenheimer approximation begins to break down and a single PES no longer provides an adequate description. An example is the study of nuclear quantum effects and electronically excited states (upper right). Finally, ML-FFs can be further enhanced by modeling additional properties. This provides direct access to a wide range of molecular spectra, building a bridge between theory and experiment (lower right). In general, such studies would be prohibitively expensive with *ab initio* methods.

by the accuracy of the underlying FF.<sup>15</sup> This is particularly problematic when polarization, or many-body interactions in general, are of significant importance, as these effects are not adequately modeled by the terms described above. While it is possible to construct polarizable FFs<sup>16–19</sup> or account for other important effects, for example, anisotropic charge distributions,<sup>20,21</sup> to improve accuracy (in exchange for computational efficiency), it is not always clear *a priori* when such modifications are necessary. Beyond that, conventional FFs require a preconceived notion of bonding patterns and thus cannot describe bond breaking or bond formation. While there exist reactive FFs offering an approximated description of reactions,<sup>22–24</sup> they are often not sufficiently accurate for quantitative studies or restricted to specific types of reactions. Mixed quantum mechanics/molecular mechanics (QM/MM) treatments<sup>25</sup> pose an alternative solution: Here, the SE is only solved for a small part of the system where high accuracy is required or reactions may take place, for example, the active site of an enzyme. Meanwhile, all remaining atoms are treated at the FF level of accuracy. Although this is more efficient than a pure quantum-mechanical approach, it is often necessary to model a large number of atoms at the QM level for converged results,<sup>26</sup> which is still highly computationally demanding.

In a “dream scenario” for computational chemists and biologists, it would be possible to treat even large systems at the highest levels of theory, which would require prohibitively large computational resources in the real world. Machine learning (ML) methods could help to achieve this dream in a principled manner, thus closing the gap between the accuracy of *ab initio* methods and the efficiency of classical FFs (Figure 1). ML methods aim to learn the functional relationship between inputs (chemical descriptors) and outputs (properties) from patterns or structure in the data. Ideally, a trained learning machine would then reflect the underlying effective “rules” of quantum mechanics.<sup>27</sup> Practically, ML models can take a shortcut by not having to solve any equations that follow from the physical laws governing the structure–property relation. Because of this unique ability, ML methods have been enjoying growing popularity in the chemical sciences in recent years. They allow to explore chemical space and predict the properties of compounds with both unprecedented efficiency and high accuracy.<sup>27–32</sup> ML has also been used to augment and accelerate traditional methods used in molecular simulations, for example, for sampling equilibrium states<sup>33,34</sup> and rare events,<sup>35</sup> computing reaction rates,<sup>36</sup> exploring protein folding dynamics<sup>37</sup> and other biophysical processes,<sup>38–42</sup> Markov state modeling,<sup>43–49</sup> and



coarse-graining<sup>50–53</sup> (for a recent review on applications of ML in molecular simulations, see ref 54). Recent advances made it even possible to predict molecular wave functions, which can act as an interface between ML and quantum chemistry,<sup>55,56</sup> as knowledge of the wave function allows to deduce many different quantum mechanical observables at once. ML can also be combined with more traditional semiempirical methods, for example by predicting accurate repulsive potentials for density functional tight-binding approaches.<sup>57</sup> Instead of circumventing equations, ML methods can also help when solving them. They have been used to find novel density functionals<sup>58–60</sup> and solutions of the Schrödinger equation.<sup>61,62</sup> Other promising applications include the generation of molecular structures to tackle inverse design problems,<sup>63–67</sup> or planning chemical syntheses.<sup>68</sup>

For constructing ML-FFs, suitable reference data to learn the relevant structure–property relation include energy, forces, or a combination of both, obtained from *ab initio* calculations. Contrary to conventional FFs, no preconceived notion of bonding patterns needs to be assumed. Instead, all chemical behavior is learned from the reference data. This allows to reconstruct the important interactions purely from atomic positions without imposing a restricted analytical form on the interatomic potential and enables a natural description of chemical reactions. For example, it is now possible to construct ML-FFs for small molecules from CCSD(T) reference data close to spectroscopic accuracy and with computational efficiency similar to conventional FFs.<sup>69,70</sup> This has enabled studies that would be prohibitively expensive with conventional methods of computational chemistry and allowed to obtain novel chemical insights (see Figure 2).

Other properties than energies and forces can be predicted as well: For example, dipole moments, which are a measure for how polar molecules are, can be used to calculate infrared spectra from MD simulations<sup>71–73</sup> and allow a comparison to experimental measurements. Other prediction targets could be used to screen large compound databases for molecules with desirable properties several orders of magnitude faster than with *ab initio* methods. The HOMO/LUMO gap, which is important for materials used in electronic devices such as solar cells,<sup>74</sup> is only one prominent example of many potentially interesting prediction targets.

This review will focus on the construction of ML-FFs for the usage in MD simulations and other applications (for details on how to set up such simulations or how to extract physical insights from them, refer to refs 75–77). The text is structured as follows: Section 2 reviews fundamental concepts of chemistry (Section 2.1) and machine learning (Section 2.2) relevant to the construction of ML-FFs and discusses special considerations when the two are combined (Section 2.3). As this article is intended for both chemists and machine learning practitioners, these sections provide all readers with the necessary background to understand the remainder of the review. Experts in either field may want to skip over the respective sections, as they discuss fundamentals. The next part (Section 3) serves as a step-by-step guide and reference for readers that want to apply ML-FFs in their own research. Here, the best practices for constructing ML-FFs are outlined, possible problems that may be encountered along the way (and how to avoid them) are discussed and an overview of several software packages, which may be used to accelerate the construction of ML-FFs, is provided. Section 4 lists several applications of ML-FFs described in the literature and

highlights physical and chemical insights made possible through the use of ML. The review is concluded in Section 5 by a discussion of obstacles that still need to be overcome to extend the applicability of ML-FFs to an even broader context.

## 2. MATHEMATICAL AND CONCEPTUAL FRAMEWORK

Section 2.1 reviews important chemical concepts such as the potential energy surface and invariance properties of physical systems, which are essential for constructing physically meaningful models. It is meant as a short summary of the most important physical principles and fundamental chemical knowledge for readers with a primarily ML-focused background who are interested in constructing ML-FFs. On the other hand, to offer readers with a chemical background a first orientation, an overview of two important methodologies in Machine Learning, namely kernel-based learning approaches and artificial neural networks, is given in Section 2.2. Finally, Section 2.3 lists constraints related to the physical invariances mentioned earlier and gives examples of models for constructing ML-FFs and how they implement these constraints in practice.

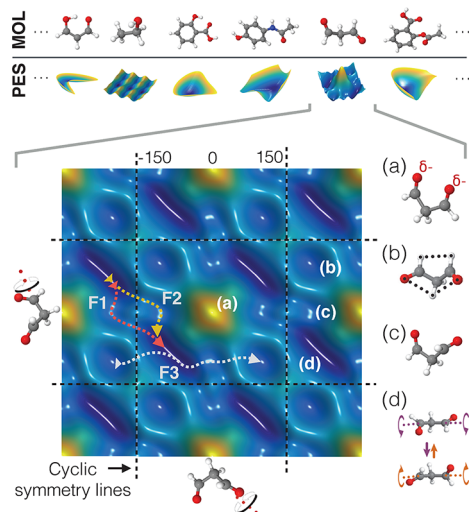
### 2.1. Chemistry Foundations

The Schrödinger equation (SE),<sup>78</sup> which describes the interaction of atomic nuclei and electrons, is sufficient for understanding most chemical phenomena and processes.<sup>10</sup> Unfortunately, it can only be solved analytically for very simple systems, such as the hydrogen atom. For more complex systems like molecules, exact numerical solutions are often impractical due to a steep increase of computational costs as a function of system size. For this reason, numerous approximation schemes have been devised to enable insights into more complicated chemical systems. Virtually all of these are based on the Born–Oppenheimer (BO) approximation,<sup>79</sup> which decouples electronic and nuclear motion so that the latter can be neglected. It is assumed that electrons adjust instantaneously to changes in the nuclear positions, which is motivated by the observation that atomic nuclei are heavier than electrons by several orders of magnitude, thus moving on a vastly different time scale. Hence, the nuclear positions appear almost stationary to the electrons and therefore enter the resulting “electronic SE” only parametrically: The energy of the electrons depends on the external potential caused by the nuclei, which in turn is fully determined by their positions and nuclear charges. By summing electronic energy and Coulomb repulsion between nuclei, the total potential energy of the system is obtained, which is one of the most important properties of molecules. Alongside entropic contributions, it determines the relative stability of different compounds, whether reactions are exothermic or endothermic, and can even serve as proxy for more complex properties. For example, the potency of some drugs can be estimated from their binding energy to biomolecules.<sup>80</sup>

**2.1.1. Potential Energy Surface.** By introducing a parametric dependency between energy and nuclei, the BO approximation implies the existence of a functional relation  $f: \{Z_i, \mathbf{r}_i\}_{i=1}^N \rightarrow E$ , which maps the nuclear charges  $Z_i$  and positions  $\mathbf{r}_i$  of  $N$  atoms directly to their potential energy  $E$ . This function, called the potential energy surface (PES), governs the dynamics of a chemical system, similar to a ball rolling on a hilly landscape. Minima (“valleys”) on the PES correspond to stable molecules and significant structural changes (or even chemical reactions) occur when a system



crosses over a transition state (“ridge”) from one minimum into another (Figure 3).



**Figure 3.** Top: Two-dimensional projections of the PESs of different molecules, highlighting rich topological differences and various possible shapes. Bottom: Cut through the PES of keto-malondialdehyde for rotations of the two aldehyde groups. Note that the shape repeats periodically for full rotations. Regions with low potential energy are drawn in blue and high energy regions in yellow. Structure (a) leads to a steep increase in energy due to the proximity of the two oxygen atoms carrying negative partial charges. Local minima of the PES are shown in (b) and (c), whereas (d) displays structural fluctuations around the global minimum. By running molecular dynamics simulations, the most common transition paths (F1, F2, and F3) between the different minima could be revealed.

Knowledge of the PES therefore also allows to predict how a system evolves over time. For example, by studying a thermal ensemble of molecules starting from the same minimum on the PES, it is possible to determine which fraction of them will reach different minima and in what time frame, allowing to assess their reactivity and which products are formed. It is also possible to deduce the macroscopic thermodynamic properties of a system by studying how it behaves at an atomic level. In such molecular dynamics (MD) simulations, a classical treatment of nuclear dynamics is sometimes sufficiently accurate. In case of significant nuclear delocalization, which may occur in systems with light atoms, strong bonds, or for shallow potential energy landscapes,<sup>81</sup> nuclear quantum effects (NQE) must be included as well. Even then, methods like path-integral MD establish a one-to-one correspondence between the properties of a quantum object and a classical system with an extended phase space, eliminating the need to solve the nuclear SE.<sup>82–84</sup>

At each time step of a dynamics simulation, the forces  $F_i$  acting on each atom  $i$  must be known so that the equations of motion can be integrated numerically (e.g., using the Verlet algorithm<sup>85</sup>). They can be derived from the PES by using the relation  $F_i = -\nabla_{\mathbf{r}_i} E$ , that is, the forces are the negative gradient of the potential energy  $E$  with respect to the atomic positions  $\mathbf{r}_i$  (see also Section 2.1.2). Forces can also be used to perform geometry optimizations, e.g., to find special configurations of atoms which correspond to critical points on the PES. For example, the height of a reaction barrier can be computed from the energy difference between the saddle point (transition

state) and either of the two minima (equilibrium structures) which are connected by it.

Although the BO approximation simplifies the SE, even approximate solutions can be computationally demanding, in particular for large systems containing many degrees of freedom. Thus, it is often unfeasible to derive *ab initio* energies and forces for each time step of an MD simulation. For this reason, analytical functions known as force fields (FFs), are typically used to represent the PES, circumventing the problem of solving an equation altogether. The difficulty is then shifted to finding an appropriate functional form and parametrization of the FF. ML methods automate this demanding and time-consuming process by learning an appropriate function from reference data.

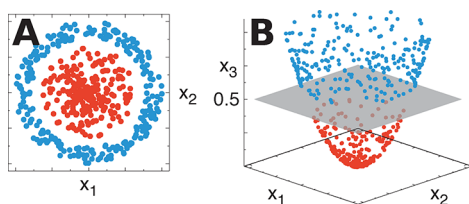
**2.1.2. Invariances of Physical Systems.** Closed physical systems are governed by various conservation laws that describe invariant properties. They are fundamental principles of nature that characterize symmetries that must not be violated. As such, conservation laws provide strong constraints that can be used as guiding principles in search of physically plausible ML models. The basic invariances of molecular systems are directly derived from Noether's theorem,<sup>86</sup> which states that each conserved quantity is associated with a differentiable symmetry of the action of a physical system. Typical conserved quantities include the total energy (following from temporal invariance) as well as angular and linear momentum (rototranslational invariance). Energy conservation imposes a particular structure on vector fields in order for them to be valid force fields with corresponding potentials. Namely, forces must be the negative gradient of the potential energy with respect to atomic positions. This relation ensures that when atoms move, they always acquire the same amount of kinetic energy as they lose in potential energy (and *vice versa*), i.e., the total energy is constant (the work done along closed paths is zero). The conservation of linear and angular momentum implies that the potential energy of a molecule only depends on the relative position of its atoms to each other and does not change with rigid rotations or translations. Another invariance (not derived from Noether's theorem) follows from the fact that, from the perspective of the electrons, atoms with the same nuclear charge appear identical to each other. They can thus be exchanged without affecting the energy and forces, which makes the PES symmetric with respect to permutations of some of its arguments. To ensure physically meaningful predictions, ML-FFs must be made invariant under the same transformations as the true PES by introducing appropriate constraints.

## 2.2. Machine Learning Foundations

A question that frequently arises for researchers new to the field of ML concerns the difference of ML modeling to plain interpolation in the noise free regression case. After all, the Shannon sampling theorem gives bounds for the number of “training samples” needed to reconstruct a band-limited signal exactly.<sup>87</sup> Since the regression tasks considered in this review use *ab initio* data as reference, they can be considered practically noise-free. Furthermore, PESs are usually smooth, which means there is a well-defined frequency cutoff in the spectrum of this “signal”. Thus, both requirements for Shannon interpolation are satisfied and it should in principle be possible to reconstruct FFs via interpolation of the training samples without error, provided there are enough of them.

This is where ML diverges from signal interpolation theory. In practice, there is often not enough data available to fully capture all the necessary information for a perfect reconstruction. In that case, the goal of ML methods is not to recover the training data, but rather to estimate the true process with its underlying regularities that also describes all new and unseen data; this is often denoted as generalization. The key to generalization is selecting a model based on the well-known principle of Occam's razor, i.e., the notion that simpler hypotheses are more likely to be correct.<sup>88</sup> The capacity of the model can be controlled using the bias–variance trade-off<sup>89</sup> (a compromise between expressiveness and complexity) and is practically done by exercising model selection techniques (see Section 2.2.3) such as cross-validation that leave out part of the data from the ML training process and use it later to obtain a valid estimate of the generalization error.<sup>30,90</sup> The reason why regularization is often needed is that ML algorithms are universal approximators that can approximate any continuous function on a closed interval arbitrarily close. Since for a finite amount of reference data infinitely many such functions are thinkable, a regularization mechanism is often needed to select a preferably simple function from the vast space of possibilities.

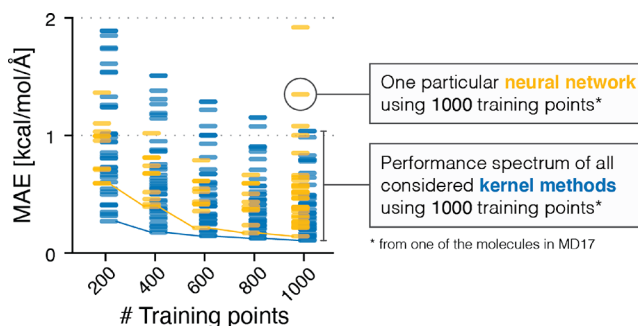
ML methods typically rely on the fact that nonlinear problems, such as predicting energy from nuclear positions, can be “linearized” by mapping the input to a (often higher-dimensional) “feature space” (see Figure 4).<sup>90–93</sup> Note that



**Figure 4.** (A) Blue and red points with coordinates  $(x_1, x_2)$  are linearly inseparable. (B) By defining a suitable mapping from the input space  $(x_1, x_2)$  to a higher-dimensional feature space  $(x_1, x_2, x_3)$ , blue and red points become linearly separable (gray plane at  $x_3 = 0.5$ ).

such feature spaces are explicitly constructed for kernel-based learning methods (see Section 2.2.1) or learned respectively for deep learning models<sup>94</sup> (see Section 2.2.2). Kernel-based methods achieve this by taking advantage of the so-called kernel trick,<sup>92,95–98</sup> which allows implicitly operating in a high-dimensional feature space without explicitly performing any computation there. In contrast, artificial neural networks (NNs) decompose a complex nonlinear function into a composition of linear transformations with learnable parameters connected by nonlinear activation functions. With increasingly many of such nonlinear transformations organized in “layers” (deep NNs), it is possible to efficiently learn highly complex feature spaces.

While NNs tend to require more training data to reach the same accuracy as kernel methods (see Figure 5),<sup>99</sup> they typically scale better to larger data sets. In general, neither method is strictly superior over the other,<sup>100</sup> and both have advantages and disadvantages that must be weighed against each other for a specific application. Recently, it has even been discovered that in the limit of infinitely wide layers, deep NNs are equivalent to kernel methods, which shifts the main differentiating factor between both methodologies to how they are constructed and trained<sup>101,102</sup> and makes deep NNs accessible to kernel-based analysis methods.<sup>103,104</sup>



**Figure 5.** Mean absolute force prediction errors (MAEs) of different ML models trained on molecules in the MD17 data set,<sup>105</sup> colored by model type. Overall, kernel methods (GDML,<sup>105</sup> sGDML,<sup>69</sup> FCHL18/19<sup>106,107</sup>) are slightly more data efficient, that is, they produce more accurate predictions with smaller training data sets, but neural network architectures (PhysNet,<sup>108</sup> SchNet,<sup>109</sup> DimeNet,<sup>110</sup> EANN,<sup>111</sup> DeePMD,<sup>112</sup> DeepPot-SE,<sup>113</sup> ACSF,<sup>114</sup> HIP-NN<sup>115</sup>) catch up quickly with increasing training set size and continue to improve when more data for training is available.

In the following, kernel methods and neural networks are described in more detail to highlight the most important properties that differentiate both methodologies.

**2.2.1. Kernel-Based Methods.** Given a data set  $\{(y_i, \mathbf{x}_i)\}_{i=1}^M$  of  $M$  reference values  $y_i \in \mathbb{R}$  for inputs  $\mathbf{x}_i \in \mathbb{R}^D$ , kernel regression aims to estimate  $y_*$  for unknown inputs  $\mathbf{x}_*$ . For example, for PES construction,  $y$  is the potential energy and  $\mathbf{x}$  encodes structural information about the atoms, i.e., their nuclear charges and relative positions in space. Popular choices for such “descriptors” are vectors of internal coordinates, Coulomb matrices,<sup>8</sup> representations of atomic environments (e.g., symmetry functions,<sup>116</sup> SOAP<sup>117</sup> or FCHL<sup>106,107</sup>), or an encoding of crystal structure.<sup>118–120</sup> See ref 121 for a recent review on structural descriptors.

The representer theorem states that the functional relation

$$y = f(\mathbf{x}) + \epsilon \quad (1)$$

where  $\epsilon$  denotes measurement noise, can be optimally approximated as a linear combination

$$f(\mathbf{x}) \approx \hat{f}(\mathbf{x}) = \sum_{i=1}^M \alpha_i K(\mathbf{x}, \mathbf{x}_i) \quad (2)$$

where  $\alpha_i$  are coefficients, and  $K(\mathbf{x}, \mathbf{x}')$  is a (typically nonlinear) symmetric and positive semidefinite function<sup>122–124</sup> that measures the similarity of two compound descriptors  $\mathbf{x}$  and  $\mathbf{x}'$  (see Figure 7). (The function  $K(\mathbf{x}, \mathbf{x}')$  computes the inner product of two points  $\phi(\mathbf{x})$  and  $\phi(\mathbf{x}')$  in some Hilbert space  $\mathcal{H}$  (the feature space) without the need to evaluate (or even know) the mapping  $\phi: \mathbb{R}^D \rightarrow \mathcal{H}$  explicitly, i.e.,  $K(\mathbf{x}, \mathbf{x}')$  is a reproducing kernel of  $\mathcal{H}$ .<sup>90,125</sup>) Examples for such functions  $K$  are the polynomial kernel

$$K(\mathbf{x}, \mathbf{x}') = (\langle \mathbf{x}, \mathbf{x}' \rangle + c)^d \quad (3)$$

where hyperparameter  $d$  is the degree of the polynomial and  $\langle \cdot, \cdot \rangle$  is the dot product, or the Gaussian kernel given by

$$K(\mathbf{x}, \mathbf{x}') = e^{-\gamma \|\mathbf{x} - \mathbf{x}'\|^2} \quad (4)$$

with hyperparameter  $\gamma$  controlling its width/scale and  $\|\cdot\|$  denoting the  $L^2$ -norm (see refs 30, 90, 97, and 126 for more examples of kernel functions).

The structure and number of dimensions of the associated Hilbert space  $\mathcal{H}$  depends on the choice of  $K(\mathbf{x}, \mathbf{x}')$  and dimension of the inputs  $\mathbf{x}$  and  $\mathbf{x}'$ . As an example, consider the polynomial kernel (eq 3) with degree  $d = 2$  and two-dimensional inputs. The corresponding homogeneous ( $c = 0$ ) polynomial mapping is given by  $\phi: (x_1, x_2) \rightarrow (x_1^2, \sqrt{2}x_1x_2, x_2^2)$ , so the associated  $\mathcal{H}$  is three-dimensional. While in this case, it is still possible to compute  $\phi$  and evaluate the inner product of two points  $\phi(\mathbf{x})$  and  $\phi(\mathbf{x}')$  explicitly, the advantage of using kernels becomes apparent when the Gaussian kernel (eq 4) is considered. Rewriting eq 4 as

$$K(\mathbf{x}, \mathbf{x}') = e^{-\gamma\|\mathbf{x}\|^2} e^{-\gamma\|\mathbf{x}'\|^2} e^{2\gamma\langle\mathbf{x}, \mathbf{x}'\rangle} \quad (5)$$

and expanding the third factor in a Taylor series  $e^{2\gamma\langle\mathbf{x}, \mathbf{x}'\rangle} = \sum_{d=0}^{\infty} \frac{1}{d!} (2\gamma\langle\mathbf{x}, \mathbf{x}'\rangle)^d$  reveals that the Gaussian kernel is equivalent to an infinite sum over (scaled) polynomial kernels (see eq 3) and the associated  $\mathcal{H}$  is  $\infty$ -dimensional. Fortunately, by using the kernel function  $K(\mathbf{x}, \mathbf{x}')$ , it is possible to operate in  $\mathcal{H}$  implicitly and evaluate  $\hat{f}(\mathbf{x})$  (eq 2) without computing the mapping  $\phi$ . This is often referred to as the kernel trick.<sup>90,92,95–97,126</sup>

It remains the question how the coefficients  $\alpha_i$  in eq 2 are determined. One way to do so is by adopting a Bayesian, or probabilistic, point of view.<sup>127,128</sup> Here, it is assumed that the reference data  $\{(y_i; \mathbf{x}_i)\}_{i=1}^M$  are generated by a Gaussian process (GP), i.e., drawn from a multivariate Gaussian distribution. For simplicity, it can be assumed that this distribution has a mean of zero, as other values can be generated by simply adding a constant term. Further, the possibility that the reference data might be contaminated by noise (for example due to uncertainties in measuring  $y_i$ ) is accounted for explicitly. Typically, Gaussian noise is assumed, i.e.,

$$y_i = f(\mathbf{x}_i) + \mathcal{N}(0, \lambda) \quad (6)$$

where  $\lambda$  is the variance of the normally distributed noise  $\mathcal{N}$ . In the GP picture, the choice of  $K(\mathbf{x}, \mathbf{x}')$  expresses an assumption about the underlying function class. For example, choosing the Gaussian kernel implies that  $f(\mathbf{x})$  does not change drastically over a length scale controlled by  $\gamma$  (see eq 4). As such, a particular kernel function  $K$  corresponds to an implicit regularization, i.e., an assumption about the underlying smoothness properties of the function to be estimated.<sup>129</sup> The challenge lies in finding a kernel that represents the structure in the data that is being modeled as good as possible.<sup>103,129</sup> Many kernels are able to approximate continuous functions on a compact subset arbitrarily well,<sup>129,130</sup> but a strong prior has the advantage of restricting the hypothesis space, which drastically improves the convergence of the learning task with respect to the available training data.<sup>131</sup>

Under these conditions, it is now possible to rigorously answer the question “given the data  $\mathbf{y} = [y_1 \cdots y_M]^T$ , how likely is it to observe the value  $y_*$  for input  $\mathbf{x}_*$ ?” As  $y_*$  is generated by the same GP as the reference data, the conditional probability  $p(y_*|\mathbf{y})$  can be expressed as

$$\begin{bmatrix} \mathbf{y} \\ y_* \end{bmatrix} \sim \mathcal{N}\left(0, \begin{bmatrix} \mathbf{K} + \lambda \mathbf{I}_M & \mathbf{K}_*^T \\ \mathbf{K}_* & K(\mathbf{x}_*, \mathbf{x}_*) \end{bmatrix}\right) \quad (7)$$

where  $\mathbf{I}_M$  is the identity matrix of size  $M$ ,  $\mathbf{K}$  is the  $M \times M$  kernel matrix<sup>90,132</sup> with entries  $K_{ij} = K(\mathbf{x}_i, \mathbf{x}_j)$  and  $\mathbf{K}_* = [K(\mathbf{x}_*, \mathbf{x}_1) \cdots K(\mathbf{x}_*, \mathbf{x}_M)]$ . In other words, eq 7 expresses a probability distribution over possible predictions, where its mean value

$$\bar{y}_* = \mathbf{K}_* (\mathbf{K} + \lambda \mathbf{I}_M)^{-1} \mathbf{y} \quad (8)$$

is the most likely estimate for  $y_*$  (given the reference data) and its variance

$$\text{var}(y_*) = K(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{K}_* (\mathbf{K} + \lambda \mathbf{I}_M)^{-1} \mathbf{K}_*^T \quad (9)$$

provides information about how strongly other likely predictions vary from the mean. Note that while eq 9 can be used as uncertainty estimate for a particular prediction, it should not be confused with error bars. The optimal coefficients  $\alpha = [\alpha_1 \cdots \alpha_M]$  in eq 2 are thus given by

$$\alpha = (\mathbf{K} + \lambda \mathbf{I}_M)^{-1} \mathbf{y} \quad (10)$$

or simply  $\alpha = \mathbf{K}^{-1} \mathbf{y}$  in the noise-free case ( $\lambda = 0$ ). However, even in the absence of noise, it can be beneficial to choose a nonzero  $\lambda$  to obtain a regularized solution. The addition of  $\lambda > 0$  to the diagonal of  $\mathbf{K}$  increases numerical stability and has the effect of damping the magnitude of the coefficients, thereby increasing the smoothness of  $\hat{f}(\mathbf{x})$ . The downside is that the known reference values  $y_i$  are only approximately reproduced. This, however, also decreases the chance of overfitting and can lead to better generalization, that is, increased accuracy when predicting unknown values.

Matrix factorization methods like Cholesky decomposition<sup>133</sup> are typically used to efficiently solve the linear problem in eq 10 in closed form. However, this type of approach scales as  $O(M^3)$  with the number of reference data and may become problematic for extremely large data sets. Iterative gradient-based solvers can reduce the complexity to  $O(M^2)$ .<sup>134</sup> Once the coefficients have been determined, the value  $y_*$  for an arbitrary input  $\mathbf{x}_*$  can be estimated according to eq 2 with  $O(M)$  complexity (a sum over all  $M$  reference data points is required).

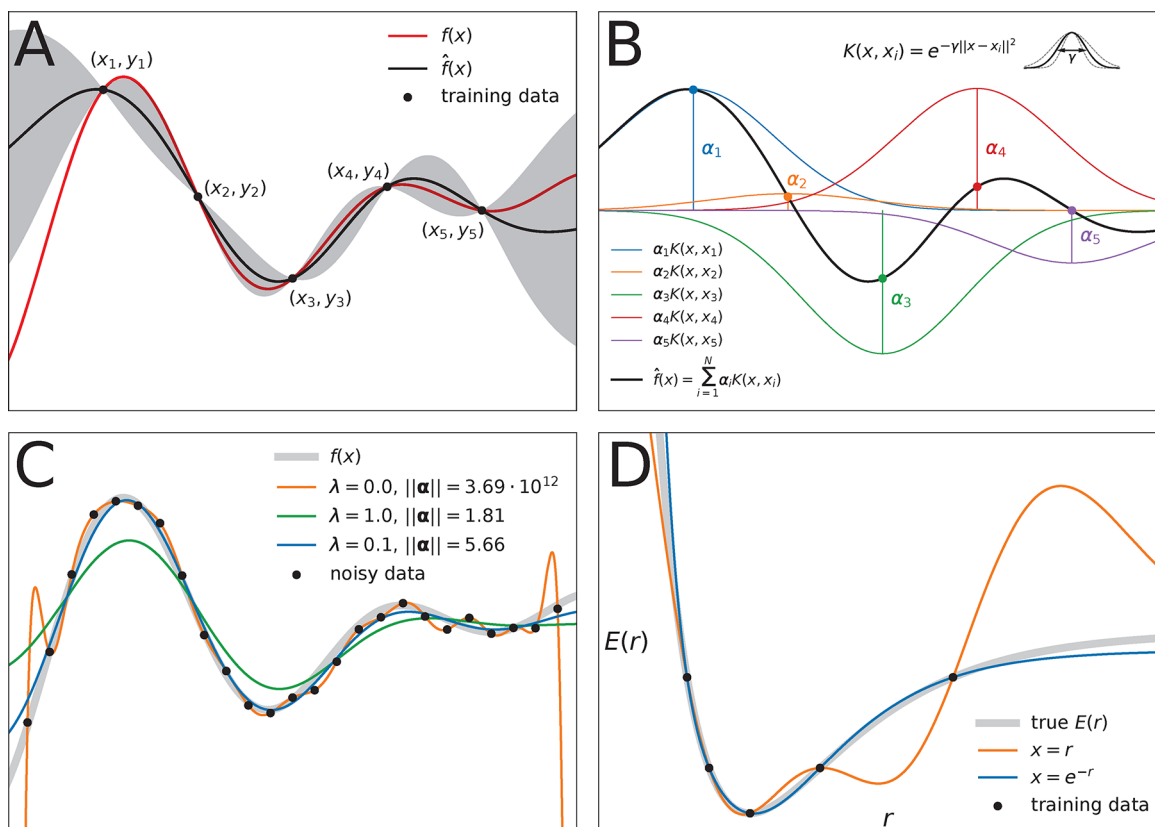
Alternatively, a variety of approximation techniques exploit that kernel matrices usually have a small numerical rank, i.e., a rapidly decaying eigenvalue spectrum. This enables approximate factorizations  $\mathbf{R}\mathbf{R}^T \approx \mathbf{K}$ , where  $\mathbf{R}$  is either a rectangular matrix  $\in \mathbb{R}^{M \times L}$  with  $L < M$  or sparse. As a result, eq 10 becomes easier to solve, albeit the result will not be exact.<sup>135–139</sup>

A straightforward approach to approximate a linear system is to pick a representative or random subset of  $L$  points  $\tilde{\mathbf{x}}$  from the data set (in principle, even arbitrary  $\tilde{\mathbf{x}}$  could be chosen) and construct a rectangular kernel matrix  $\mathbf{K}_{LM} \in \mathbb{R}^{L \times M}$  with entries  $K_{LM,ij} = K(\tilde{\mathbf{x}}_i, \mathbf{x}_j)$ . Then the corresponding coefficients can be obtained via the Moore-Penrose pseudoinverse:<sup>140,141</sup>

$$\tilde{\alpha} = (1 + \lambda)^{-1} (\mathbf{K}_{LM} \mathbf{K}_{LM}^T)^{-1} \mathbf{K}_{LM} \mathbf{y} \quad (11)$$

Solving eq 11 scales as  $O(ML^2)$  and is much less computationally demanding than inverting the original matrix in eq 10. Once the  $L$  coefficients  $\tilde{\alpha}$  are obtained, the model can be evaluated with  $\hat{f}(\mathbf{x}) = \sum_L \tilde{\alpha}_i K(\mathbf{x}, \tilde{\mathbf{x}}_i)$ , i.e., an additional benefit is that evaluation now scales as  $O(L)$  instead of  $O(M)$  (see eq 2).





**Figure 6.** Overview of the mathematical concepts that form the basis of kernel methods. (A) Gaussian process regression of a one-dimensional function  $f(x)$  (red line) from  $M = 5$  data samples  $(x_i, y_i)$ . The black line  $\hat{f}(x)$  depicts the mean (eq 8) of the conditional probability  $p(y_*|y)$  (see eq 7), whereas the gray area depicts two standard deviations from its mean (see eq 9). Note that predictions are most confident in regions where training data is present. (B) Function  $\hat{f}(x)$  can be expressed as a linear combination of  $M$  kernel functions  $K(x, x_i)$  weighted with regression coefficients  $\alpha_i$  (see eq 2). In this example, the Gaussian kernel (eq 4) is used (the hyperparameter  $\gamma$  controls its width). (C) Influence of noise on prediction performance. Here, the function  $f(x)$  (thick gray line) is learned from  $M = 25$  samples, however, each data point  $(x_i, y_i)$  contains observational noise (see eq 6). When the coefficients  $\alpha_i$  are determined without regularization, i.e., no noise is assumed to be present, the model function reproduces the training samples faithfully, but undulates wildly between data points (orange line,  $\lambda = 0$ ). The regularized solution (blue line,  $\lambda = 0.1$ , see eq 10) is much smoother and stays closer to the true function  $f(x)$ , but individual data points are not reproduced exactly. When the regularization is too strong (green line,  $\lambda = 1.0$ ), the model function becomes unable to fit the data. Note how regularization shrinks the magnitude of the coefficient vectors  $\|\alpha\|$ . (D) For constructing force fields, it is necessary to encode molecular structure with a representation  $x$ . The choice of this structural descriptor may strongly influence model performance. Here, the potential energy  $E$  of a diatomic molecule (thick gray line) is learned from  $M = 5$  data points by two kernel machines using different structural representations (both models use a Gaussian kernel). When the interatomic distance  $r$  is used as descriptor (orange line,  $x = r$ ), the predicted potential energy oscillates between data points, leading to spurious minima and qualitatively wrong behavior for large  $r$ . A model using the descriptor  $x = e^{-r}$  (blue line) predicts a physically meaningful potential energy curve that is qualitatively correct even when the model extrapolates.

However, the approximation above gives rise to an overdetermined system with fewer parameters than training points and therefore reduced model capacity. Strictly speaking, the involved matrix does not satisfy the properties of a kernel matrix anymore, as it is neither symmetric nor positive semidefinite. To obtain a kernel matrix that still maintains these properties, the Nyström<sup>135</sup> approximation

$$\mathbf{K} \approx \tilde{\mathbf{K}} = \mathbf{K}_{LM}^T \mathbf{K}_{LL}^{-1} \mathbf{K}_{LM} \quad (12)$$

can be used instead. Here, the submatrix  $\mathbf{K}_{LL}$  is a true kernel matrix between all inducing points  $\tilde{x}_i$ . Using the Woodbury matrix identity,<sup>142</sup> the regularized inverse is given by

$$(\tilde{\mathbf{K}} + \lambda \mathbf{I}_M)^{-1} = \lambda^{-1} [\mathbf{I}_M - \mathbf{K}_{LM}^T (\lambda \mathbf{K}_{LL} + \mathbf{K}_{LM} \mathbf{K}_{LM}^T)^{-1} \mathbf{K}_{LM}] \quad (13)$$

and  $\tilde{\alpha} = (\tilde{\mathbf{K}} + \lambda \mathbf{I}_M)^{-1} y$ . The computational complexity of solving the Nyström approximation is  $O(L^3 + ML^2)$

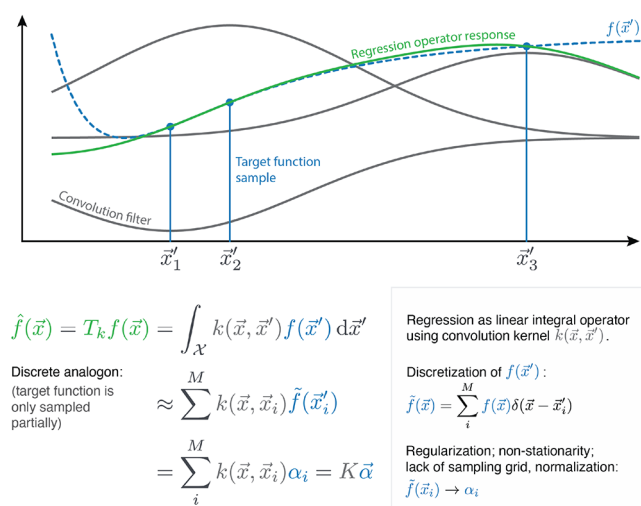
It should be mentioned that kernel regression methods are known under different names in the literature of different communities. Because of their relation to GPs, some authors prefer the name Gaussian process regression (GPR). Others favor the term kernel ridge regression (KRR), since determining the coefficients with eq 10 corresponds to solving a least-squares objective with  $L^2$ -regularization in the kernel feature space  $\phi$  and is similar to ordinary ridge regression.<sup>143</sup> Sometimes, the method is also referred to as reproducing kernel Hilbert space (RKHS) interpolation, since eq 2 “interpolates” between known reference values (when coefficients are determined with  $\lambda = 0$ , all known reference values are reproduced exactly). All these methods are formally equivalent and essentially differ only in the manner the relevant equations are derived. There are small philosophical differences, however: For example, in the KRR and RKHS pictures,  $\lambda$  in eq 10 is a regularization hyperparameter that has to be introduced *ad hoc*, whereas in the GPR picture,  $\lambda$  is directly

related to the Gaussian noise in eq 6. The expansion coefficients obtained from eq 10 can change drastically depending on the choice of  $\lambda$ , so this is an important detail. Further, while eq 9 can be used to compute uncertainty estimates for all kernel regression methods, the GPR picture allows to relate it directly to the variance of a Gaussian process.

The most important concepts discussed in this section are summarized visually in Figure 6.

#### Regression as linear integral operator

Machine learning: Interpolation + Regularization = Generalization



**Figure 7.** Kernel ridge regression can be understood as a linear integral operator  $T_k$  that is applied to the (only partially known) target function of interest  $f(\mathbf{x})$ . Such operators are defined as convolutions with a continuous kernel function  $K$ , whose response is the regression result. Because the training data is typically not sampled on a grid, this convolution task transforms to a linear system that yields the regression coefficients  $\alpha$ . Because only  $T_k f(\mathbf{x})$  and not the true  $f(\mathbf{x})$  is recovered, the challenge is to find a kernel that defines an operator that leaves the relevant parts of its original function invariant. This is why the Gaussian kernel (eq 4) is a popular choice: Depending on the chosen length scale  $\gamma$ , it attenuates high frequency components, while passing through the low frequency components of the input, therefore making only minimal assumptions about the target function. However, stronger assumptions (e.g., by combining kernels with physically motivated descriptors) increase the sample efficiency of the regressor.

**2.2.2. Artificial Neural Networks.** Originally, artificial neural networks (NNs) were, as suggested by their name, intended to model the intricate networks formed by biological neurons.<sup>144</sup> Since then, they have become a standard ML algorithm<sup>94,98,144–150</sup> only remotely related to their original biological inspiration. In the simplest case, the fundamental building blocks of NNs are dense (or “fully-connected”) layers—linear transformations from input vectors  $\mathbf{x} \in \mathbb{R}^{n_{\text{in}}}$  to output vectors  $\mathbf{y} \in \mathbb{R}^{n_{\text{out}}}$  according to

$$\mathbf{y} = \mathbf{W}\mathbf{x} + \mathbf{b} \quad (14)$$

where both weights  $\mathbf{W} \in \mathbb{R}^{n_{\text{out}} \times n_{\text{in}}}$  and biases  $\mathbf{b} \in \mathbb{R}^{n_{\text{out}}}$  are parameters, and  $n_{\text{in}}$  and  $n_{\text{out}}$  denote the number of dimensions of  $\mathbf{x}$  and  $\mathbf{y}$ , respectively. Evidently, a single dense layer can only express linear functions. Nonlinear relations between inputs and outputs can only be modeled when at least two dense layers are stacked and combined with a nonlinear activation function  $\sigma$ :

$$\mathbf{h} = \sigma(\mathbf{W}\mathbf{x} + \mathbf{b}),$$

$$\mathbf{y} = \mathbf{W}'\mathbf{h} + \mathbf{b}' \quad (15)$$

Provided that the number of dimensions of the “hidden layer”  $\mathbf{h}$  is large enough, this arrangement can approximate any mapping between inputs  $\mathbf{x}$  and outputs  $\mathbf{y}$  to arbitrary precision, i.e., it is a general function approximator.<sup>151,152</sup>

In theory, shallow NNs as shown above are sufficient to approximate any functional relationship.<sup>152</sup> However, deep NNs with multiple hidden layers are often superior and were shown to be more parameter-efficient.<sup>153–156</sup> To construct a deep NN,  $L$  hidden layers are combined sequentially

$$\mathbf{h}_1 = \sigma(\mathbf{W}_1\mathbf{x} + \mathbf{b}_1),$$

$$\mathbf{h}_2 = \sigma(\mathbf{W}_2\mathbf{h}_1 + \mathbf{b}_2),$$

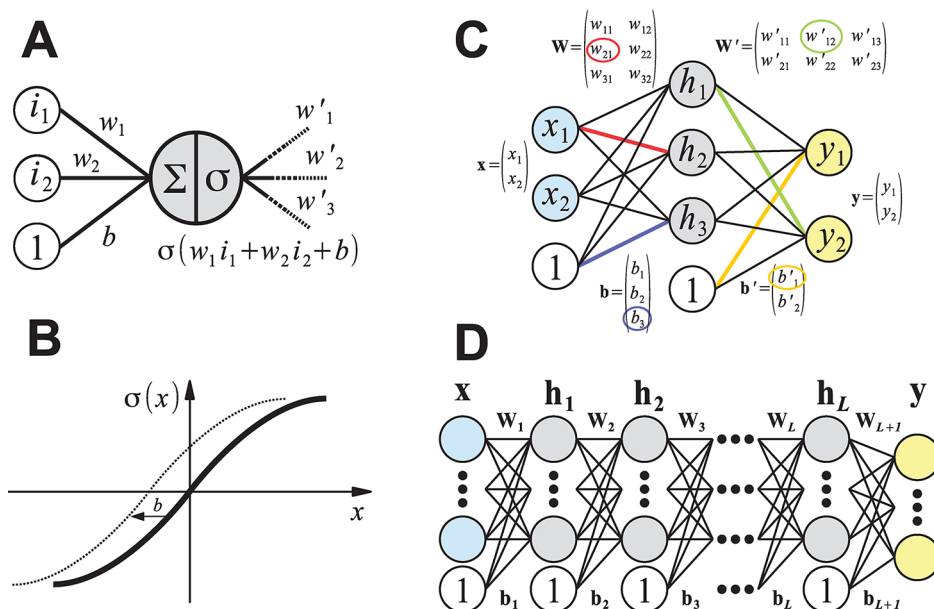
$$\vdots$$

$$\mathbf{h}_L = \sigma(\mathbf{W}_L\mathbf{h}_{L-1} + \mathbf{b}_L),$$

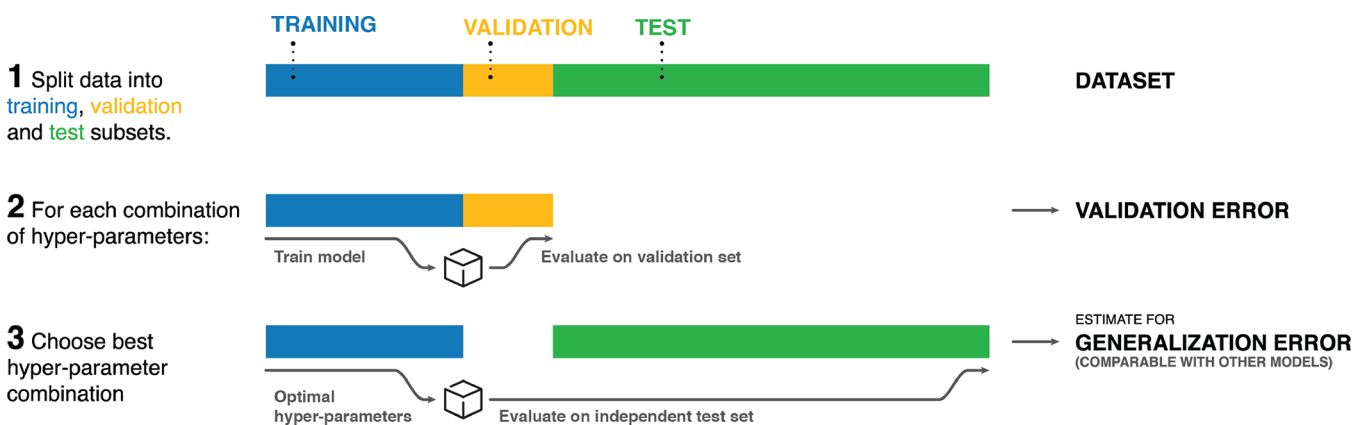
$$\mathbf{y} = \mathbf{W}_{L+1}\mathbf{h}_L + \mathbf{b}_{L+1} \quad (16)$$

mapping the input  $\mathbf{x}$  to several intermediate feature representations  $\mathbf{h}_l$ , until the output  $\mathbf{y}$  is obtained by a linear regression on the features  $\mathbf{h}_L$  in the final layer. For PES construction, typically, the NN maps a representation of chemical structure  $\mathbf{x}$  to a one-dimensional output representing the energy. Contrary to the coefficients  $\alpha$  in kernel methods (see eq 10), the parameters  $\{\mathbf{W}_l\mathbf{b}_l\}_{l=1}^{L+1}$  of an NN cannot be fitted in closed form. Instead, they are initialized randomly and optimized (usually using a variant of stochastic gradient descent) to minimize a loss function that measures the discrepancy between the output of the NN and the reference data.<sup>157</sup> A common choice is the mean squared error (MSE), which is also used in kernel methods. During training, the loss and its gradient are estimated from randomly drawn batches of training data, making each step independent of the number of training data  $M$ . On the other hand, finding the coefficients for kernel methods scales as  $O(M^3)$  due to the need of inverting the  $M \times M$  kernel matrix. Evaluating an NN according to eq 16 for a single input  $\mathbf{x}$  scales linearly with respect to the number of model parameters. The same is true for kernel methods, but here the number of model parameters is tied to the number of reference data  $M$  used for training the model (see eq 2), which means that evaluating kernel methods scales  $O(M)$ . As the evaluation cost of NNs is independent of  $M$  and only depends on the chosen architecture, they are typically the method of choice for learning large data sets. A schematic overview of the mathematical concepts behind NNs is given in Figure 8.

**2.2.3. Model Selection: How to Choose Hyperparameters.** In addition to the parameters that are determined when learning an ML model for a given data set, for example, the weights  $\mathbf{W}$  and biases  $\mathbf{b}$  in NNs or the regression coefficients  $\alpha$  in kernel methods, many models contain hyperparameters that need to be chosen before training. They allow to tune a given model to the prior beliefs about the data set/underlying physics and thus play a significant role in how a model generalizes to different data patterns. Two types of hyperparameters can be distinguished. The first kind influences the composition of the model itself, such as the type of kernel or the NN architecture, whereas the second kind affects the training procedure and thus the final parameters of the trained model. Examples for hyperparameters are the width (number of neurons per layer) and



**Figure 8.** Schematic representation of the mathematical concepts underlying artificial (feed-forward) neural networks. (A) A single artificial neuron can have an arbitrary number of inputs and outputs. Here, a neuron that is connected to two inputs  $i_1$  and  $i_2$  with “synaptic weights”  $w_1$  and  $w_2$  is depicted. The bias term  $b$  can be thought of as the weight of an additional input with a value of 1. Artificial neurons compute the weighted sum of their inputs and pass this value through an activation function  $\sigma$  to other neurons in the neural network (here, the neuron has three outputs with connection weights  $w'_1$ ,  $w'_2$ , and  $w'_3$ ). (B) Possible activation function  $\sigma(x)$ . The bias term  $b$  effectively shifts the activation function along the  $x$ -axis. Many nonlinear functions are valid choices, but the most popular are sigmoid transformations such as  $\tanh(x)$  or (smooth) ramp functions, for example,  $\max(0, x)$  or  $\ln(1 + e^x)$ . (C) Artificial neural network with a single hidden layer of three neurons (gray) that maps two inputs  $x_1$  and  $x_2$  (blue) to two outputs  $y_1$  and  $y_2$  (yellow), see eq 15. For regression tasks, the output neurons typically use no activation function. Computing the weighted sums for the neurons of each layer can be efficiently implemented as a matrix vector product (eq 14). Some entries of the weight matrices ( $W$  and  $W'$ ) and bias vectors ( $b$  and  $b'$ ) are highlighted in color with the corresponding connection in the diagram. (D) Schematic depiction of a deep neural network with  $L$  hidden layers (eq 16). Compared to using a single hidden layer with many neurons, it is usually more parameter-efficient to connect multiple hidden layers with fewer neurons sequentially.



**Figure 9.** Overview of model selection process.

depth (number of hidden layers) of an NN, the kernel width  $\gamma$  (see eq 4), or the strength of regularization terms (e.g.,  $\lambda$  in eq 10).

The range of valid values is strongly dependent on the hyperparameter in question. For example, certain hyperparameters might need to be selected from the positive real numbers (e.g.,  $\gamma$  and  $\lambda$ , see above), while others are restricted to positive integers or have interdependencies (such as depth and width of an NN). This is why hyperparameters are often optimized with primitive exhaustive search schemes like grid search or random search in combination with educated guesses for suitable search ranges, or more sophisticated Bayesian approaches.<sup>158</sup> Common gradient-based optimization methods

can typically not be applied effectively. Fortunately, for many hyperparameters, model performance is fairly robust to small changes and good default values can be determined, which work across many different data sets.

Before any hyperparameters may be optimized, a so-called test set must be split off from the available reference data and kept strictly separate. The remainder of the data is further divided into a training and a validation set. This is done because the performance of ML models is not judged by how well they predict the data they were trained on, as it is often possible to achieve arbitrarily small errors in this setting. Instead, the generalization error, that is, how well the model is able to predict unseen data, is taken as indicator for the quality



of a model. For this reason, for every trial combination of hyperparameters, a model is trained on the training data and its performance measured on the validation set to estimate the generalization error. Finally, the best performing model is selected. To get better statistics for estimates of the generalization error, instead of splitting the remaining data (reference data excluding test set) into just two parts, it is also possible to divide it into  $k$  parts (or folds). Then a total of  $k$  models is trained, each using  $k - 1$  folds as the training set and the last fold as validation set. This method is known as  $k$ -fold cross-validation.<sup>30,159</sup>

As the validation data influence model selection (even though it is not used directly in the training process), the validation error may give too optimistic estimates and is no reliable way to judge the true generalization error of the final model. A more realistic value can be obtained by evaluating the model on the held-out test set, which has neither direct nor indirect influence on model selection. To not invalidate this estimate, it is crucial not to further tweak any parameters or hyperparameters in response to test set performance. More details on how to construct ML models (including the selection of hyperparameters and the importance of keeping an independent test set) can be found in Section 3. The model selection process is summarized in Figure 9.

### 2.3. Combining Machine Learning and Chemistry

The need for ML methods often arises from the lack of theory to describe a desired mapping between input and output. A classical example for this is image classification: It is not clear how to distinguish between pictures of different objects, as it is unfeasible to formulate millions of rules by hand to solve this task. Instead, the best results are currently achieved by learning statistical image characteristics from hundreds of thousands of examples that were extracted from a large data set representing a particular object class. From that, the classifier learns to estimate the distribution inherent to the data in terms of feature extractors with learned parameters like convolution filters that reflect different scales of the image statistics.<sup>94,98,101</sup> This working principle represents the best approach known to date to tackle this particular challenge in the field of computer vision.

On the other hand, the benchmark for solving molecular problems is set by rigorous physical theory that provides essentially exact descriptions of the relationships of interest. While the introduction of approximations to exact theories is common practice and essential to reduce their complexity to a workable level, those simplifications are always physical or mathematical in nature. This way, the generality of the theory is only minimally compromised, albeit with the inevitable consequence of a reduction in predictive power. In contrast, statistical methods can be essentially exact, but only in a potentially very narrow regime of applicability. Thus, a main role of ML algorithms in the chemical sciences has been to shortcut some of the computational complexity of exact methods by means of empirical inference, as opposed to providing some mapping between input and output at all (as is the case for image classification). Notably, recent developments could show that machine learning can provide novel insight beyond providing efficient shortcuts of complex physical computations.<sup>33,55,59,62,70,105,160,161</sup>

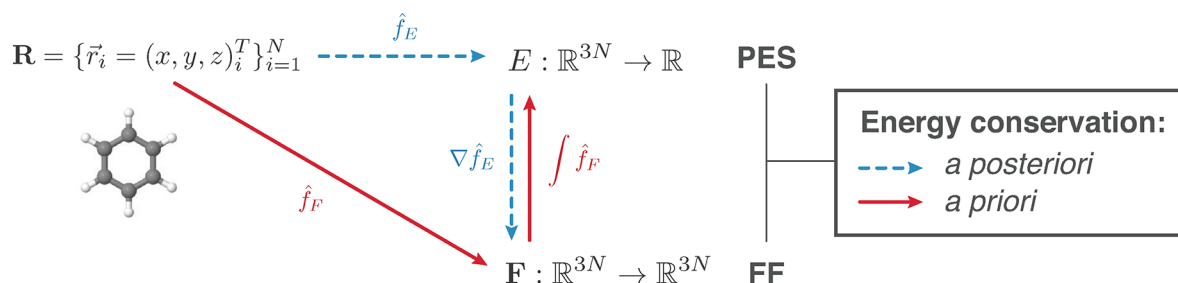
Force field construction poses unique challenges that are absent from traditional ML application domains, as much more stringent demands on accuracy are placed on ML approaches

that attempt to offer practical alternatives to established methods. Additionally, considerable computational cost is associated with the generation of high-level *ab initio* training data, with the consequence that practically obtainable data sets with sufficiently high quality are typically not very large. This is in stark contrast with the abundance of data in traditional ML application domains, such as computer vision, natural language processing etc. The challenge in chemistry, however, is to retain the generality, generalization ability and versatility of ML methods, while making them accurate, data-efficient, transferable, and scalable.

**2.3.1. Physical Constraints.** To increase data efficiency and accuracy, ML-FFs can (and should) exploit the invariances of physical systems (see Section 2.1.2), which provide additional information in ways that are not directly available for other ML problems. Those invariances can be used to reduce the function space from which the model is selected, in this manner effectively reducing the degrees of freedom for learning,<sup>69,162</sup> making the learning problem easier and thus also solvable with a fraction of data. As ML algorithms are universal approximators with virtually no inherent flexibility restrictions, it is important that physically meaningful solutions are obtained. In the following, important physical constraints of such solutions and possible ways of their realization are discussed in detail. Furthermore, existing kernel-based methods and neural network architectures tailored for the construction of FFs and how they implement these physical constraints in practice are described.

**2.3.1.1. Energy Conservation.** A necessary requirement for ML-FFs is that, in the absence of external forces, the total energy of a chemical system is conserved (see Section 2.1.2). When the potential energy is predicted by any differentiable method and forces derived from its gradient, they will be conservative by construction. However, when forces are predicted directly, this is generally not true, which makes deriving energies from force samples slightly more complicated. The main challenge to overcome is that not every vector field is necessarily a valid gradient field. Therefore, the learning problem cannot simply be cast in terms of a standard multiple output regression task, where the output variables are modeled without enforcing explicit correlations. A big advantage of predicting forces directly is that they are true quantum-mechanical observables within the BO approximation by virtue of the Hellmann–Feynman theorem,<sup>163,164</sup> i.e., they can be calculated analytically and therefore at a relatively low additional cost when generating *ab initio* reference data. As a rough guideline, the computational overhead for analytic forces scales with a factor of only around  $\sim 1$ – $7$  on top of the energy calculation.<sup>165</sup> In contrast, at least  $3N + 1$  energy evaluations would be necessary for a numerical approximation of the forces by using finite differences. For example, at the PBE0/DFT (density functional theory with the Perdew–Burke–Ernzerhof hybrid functional) level of theory,<sup>166</sup> calculating energy and analytical forces for an ethanol molecule takes only  $\sim 1.5$  times as long as calculating just the energy (the exact value is implementation-dependent), whereas for numerical gradients, a factor of at least  $\sim 10$  would be expected.

As forces provide additional information about how the energy changes when an atom is moved, they offer an efficient way to sample the PES, which is why it is desirable to formulate ML models that can make direct use of them in the training process. Another benefit of a direct reconstruction of the forces is that it avoids the amplification of estimation errors



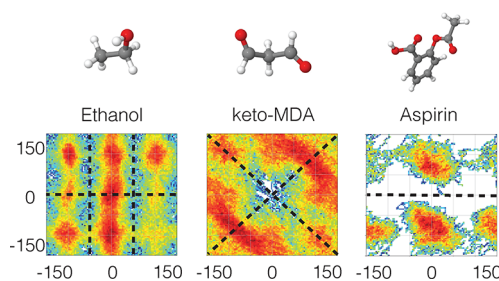
**Figure 10.** Differentiation of an energy estimator (blue) versus direct force reconstruction (red). The law of energy conservation is trivially obeyed in the first case but requires explicit *a priori* constraints in the latter scenario. The challenge in estimating forces directly lies in the complexity arising from their high 3N-dimensionality (three force components for each of the  $N$  atoms) in contrast to predicting a single scalar for the energy.

due to the derivative operator that would otherwise be applied to the PES reconstruction (see Figure 10).<sup>58,70,105</sup>

**2.3.1.2. Rototranslational Invariance.** A crucial requirement for ML-FFs is the rotational and translational invariance of the potential energy, i.e.,  $E(\mathbf{R}) = E(\mathbf{R}\mathbf{R} + \mathcal{T})$ , where  $\mathcal{R}$  and  $\mathcal{T}$  are rigid rotations and translations and  $\mathbf{R}$  are the Cartesian coordinates of the atoms. As long as the representation  $\mathbf{x}(\mathbf{R})$  of chemical structure chosen as input for the ML model itself is rototranslationally invariant, ML-FFs inherit its desired properties and even the gradients will automatically behave in the correct equivariant way due to the outer derivative  $\partial(\mathbf{x}(\mathbf{R}\mathbf{R} + \mathcal{T})/\partial\mathbf{R} = \mathcal{R}\partial\mathbf{x}(\mathbf{R})/\partial\mathbf{R}$ . One example of appropriate features to construct a representation  $\mathbf{x}$  with the desired properties are pairwise distances. For a system with  $N$  atoms, there are  $\binom{N}{2}$  different pairwise distances that result in reasonably sized feature sets for systems with a few dozen atoms. Apart from very few pathological cases, this representation is complete, in the sense that any possible configuration of the system can be described exactly and uniquely.<sup>117</sup> However, while pairwise distances serve as an efficient parametrization of some geometry distortions like bond stretching, they are relatively inefficient in describing others, for example, rotations of functional groups. In the latter case, many distances are affected even for slight angular changes, which can pose a challenge when trying to learn the geometry-energy mapping. Complex transition paths or reaction coordinates are often better described in terms of bond and torsion angles in addition to pairwise distances. The problem is that the number of these features grows rather quickly, with  $\binom{N}{3}$  and  $\binom{N}{4}$ , respectively. At that rate, the size of the feature set quickly becomes a bottleneck, resulting in models that are slow to train and evaluate. While an expert choice of relevant angles would circumvent this issue, it reduces some of the “data-driven” flexibility that ML models are typically appreciated for. Note that models without rototranslational invariance are practically unusable, as they may start to generate spurious linear or angular momentum during dynamics simulations.

**2.3.1.3. Indistinguishability of Identical Atoms.** In the BO approximation, the potential energy of a chemical system only depends on the charges and positions of the nuclei. As a consequence, the PES is symmetric under permutation of atoms with the same nuclear charge. However, symmetric regions are not necessarily sampled in an unbiased way during MD simulations (see Figure 11). Consequently, ML-FFs that are not constrained to treat all symmetries equivalently may

predict different results when permuting atoms (due to the uneven sampling).



**Figure 11.** Regions of the PESs for ethanol, keto-malondialdehyde and aspirin visited during a 200 ps *ab initio* MD simulation at 500 K using the PBE+TS/DFT level of theory<sup>167,168</sup> (density functional theory with the Perdew–Burke–Ernzerhof functional and Tkatchenko–Scheffler dispersion correction). The black dashed lines indicate the symmetries of the PES. Note that regions related by symmetry are not necessarily visited equally often.

While it is in principle possible to arrive at a ML-FF that is symmetric with respect to permutations of same-species atoms indirectly via data augmentation<sup>29,169</sup> or by simply using data sets that naturally include all relevant symmetric configurations in an unbiased way, there are obvious scaling issues with this approach. It is much more efficient to impose the right constraints onto the functional form of the ML-FF such that all relevant symmetric variants of a certain atomic configuration appear equivalent automatically. Such symmetric functions can be constructed in various ways, each of which has advantages and disadvantages.

Assignment-based approaches do not symmetrize the ML-FF *per se*, but instead aim to normalize its input, such that all symmetric variants of a configuration are mapped to the same internal representation. In its most basic realization, this assignment is done heuristically, that is, by using inexact, but computationally cheap criteria. Examples for this approach are the Coulomb matrix<sup>28</sup> or the Bag-of-Bonds<sup>31</sup> descriptors, that use simple sorting schemes for this purpose. Histograms<sup>107,170</sup> and some density-based<sup>117,171,172</sup> approaches follow the same principle, although not explicitly. All of those schemes have in common that they compare the features in aggregate as opposed to individually. A disadvantage is that dissimilar features are likely to be compared to each other or treated as the same, which limits the accuracy of the prediction. Such weak assignments are better suited for data sets with diverse conformations rather than gathered from MD trajectories that contain many similar geometries. In the latter case, the

assignment of features might change as the geometry evolves, which would lead to discontinuities in the prediction and would effectively be treated by the ML model as noise (see *eq 1*).

An alternative path is to recover the true correspondence of molecular features using a graph matching approach.<sup>173,174</sup> Each input  $\mathbf{x}$  is matched to a canonical permutation of atoms  $\tilde{\mathbf{x}} = \mathbf{P}\mathbf{x}$  before generating the prediction. This procedure effectively compresses the PES to one of its symmetric subdomains (see dashed black lines in *Figure 11*), but in an exact way. Note that graph matching is in all generality an NP-complete problem which can only be solved approximately. In practice, however, several algorithms exist to ensure at least consistency in the matching process if exactness can not be guaranteed.<sup>175</sup> A downside of this strategy is that any input must pass through a matching process, which is relatively costly, despite being approximate. Another issue is that the boundaries of the symmetric subdomains of the PES will necessarily lie in the extrapolation regime of the reconstruction in which prediction accuracy tends to degrade. As the molecule undergoes symmetry transformations, these boundaries are frequently crossed, to the detriment of prediction performance.

Arguably the most universal way of imposing symmetry, especially if the functional form of the model is already given, is via invariant integration over the relevant symmetry group  $f_{\text{sym}}(\mathbf{x}) = \int_{\pi \in S} f(\mathbf{P}_{\pi} \mathbf{x})$ . Typically,  $S$  would be the permutation group and  $\mathbf{P}_{\pi}$  the corresponding permutation matrix that transforms each vector of atom positions  $\mathbf{x}$ . Some approaches<sup>117,176,177</sup> avoid this implicit ordering of atoms in  $\mathbf{x}$  by adopting a three-dimensional density representation of the molecular geometry defined by the atom positions, albeit at the cost of losing rotational invariance, which then must be recovered by integration. Invariant integration gives rise to functional forms that are truly symmetric and do not require any pre- or postprocessing of the in- and output data. A significant disadvantage is, however, that the cardinality of even basic symmetry groups is exceedingly high, which affects both training and prediction times.

This combinatorial challenge can be solved by limiting the invariant integral to the physical point group and fluxional symmetries that actually occur in the training data set. Such a subgroup of meaningful symmetries can be automatically recovered and is often rather small.<sup>165</sup> For example, each of the molecules benzene, toluene and azobenzene have only 12 physically relevant symmetries, whereas their full symmetric groups have orders  $6!6!$ ,  $7!8!$ , and  $12!10!2!$  symmetries, respectively.

**2.3.2. (Symmetric) Gradient Domain Machine Learning ((s)GDML).** Gradient domain machine learning (GDML) is a kernel-based method introduced as a data efficient way to obtain accurate reconstructions of flexible molecular force fields from small reference data sets of high-level *ab initio* calculations.<sup>105</sup> Contrary to most other ML-FFs, instead of predicting the energy and obtaining forces by derivation with respect to nuclear coordinates, GDML predicts the forces directly. As mentioned in *Section 2.3.1*, forces obtained in this way may violate energy conservation. To ensure conservative forces, the key idea is to use a kernel  $\mathbf{K}(\mathbf{x}, \mathbf{x}') = \nabla_{\mathbf{x}} K_E(\mathbf{x}, \mathbf{x}') \nabla_{\mathbf{x}'}^T$  that models the forces  $\mathbf{F}$  as a transformation of an unknown potential energy surface  $E$  such that

$$\mathbf{F} = -\nabla E \approx \mathcal{GP}[-\nabla \mu_E(\mathbf{x}), \nabla_{\mathbf{x}} K_E(\mathbf{x}, \mathbf{x}') \nabla_{\mathbf{x}'}^T] \quad (17)$$

Here,  $\mu_E: \mathbb{R}^D \rightarrow \mathbb{R}$  and  $K_E: \mathbb{R}^D \times \mathbb{R}^D \rightarrow \mathbb{R}$  are the prior mean and covariance functions of the latent energy-based Gaussian process  $\mathcal{GP}$ , respectively. The descriptor of chemical structure  $\mathbf{x} \in \mathbb{R}^D$  consists of the inverse of all  $D$  pairwise distances, which guarantees rototranslational invariance of the energy. Training on forces is motivated by the fact that they are available analytically from electronic structure calculations, with only moderate computational overhead atop energy evaluations. The big advantage is that for a training set of size  $M$ , only  $M$  reference energies are available, whereas there are three force components for each of the  $N$  atoms, that is, a total of  $3NM$  force values. This means that a kernel-based model trained on forces contains more coefficients (see *eq 2*) and is thus also more flexible than an energy-based variant. Additionally, the amplification of noise due to the derivative operator is avoided.

A limitation of the GDML method is that the structural descriptor  $\mathbf{x}$  is not permutationally invariant because the values of its entries (inverse pairwise distances) change when atoms are reordered. An extension of the original approach, sGDML<sup>69,165</sup> (symmetric GDML), additionally incorporates all relevant rigid space group symmetries, as well as dynamic nonrigid symmetries of the system at hand into the kernel, to further improve its efficiency and ensure permutational invariance. Usually, the identification of symmetries requires chemical and physical intuition about the system at hand, which is impractical in an ML setting. Here, however, a data-driven multipartite matching approach is employed to automatically recover permutations of atoms that appear within the training set.<sup>165</sup> A matching process finds permutation matrices  $\mathbf{P}$  that realize the assignment between adjacency matrices  $(\mathbf{A})_{ij} = \|\vec{r}_i - \vec{r}_j\|$  of molecular graph pairs  $G$  and  $H$  in different energy states

$$\arg \min_{\tau} \mathcal{L}(\tau) = \|\mathbf{P}(\tau) \mathbf{A}_G \mathbf{P}(\tau)^T - \mathbf{A}_H\|^2 \quad (18)$$

and thus between symmetric transformations.<sup>178</sup> The resulting approximate local pairwise assignments are subsequently globally synchronized using transitivity as the consistency criterion<sup>175</sup> to eliminate impossible assignments. By limiting this search to the training set, combinatorially feasible, but physically irrelevant permutations  $\tau$  are automatically excluded (ones that are inaccessible without crossing impassable energy barriers). Such hard symmetry constraints (derived from the training set) greatly reduce the intrinsic complexity of the learning problem without biasing the estimator since no additional approximations are introduced.

**2.3.3. Gaussian Approximation Potentials (GAPs).** Gaussian approximation potentials (GAPs)<sup>179</sup> were originally developed for materials such as bulk crystals, but were later also applied to molecules.<sup>180</sup> They scale linearly with the number of atoms of a system and can accommodate for periodic boundary conditions. Similar to high-dimensional neural network potentials<sup>114</sup> (see *Section 2.3.4*), GAPs decompose each system into atom-centered environments  $i$  such that its energy can be written as the sum of atomic contributions

$$E = \sum_{i=1}^N E_i(\{\mathbf{r}_{ij}\}_{j \in [1, N]}) \quad (19)$$



with  $\mathbf{r}_{ij} = \mathbf{r}_j - \mathbf{r}_i$  and  $\mathbf{r}_i$  being the position of atom  $i$ . A smooth cutoff function is applied to the pairwise distances  $\|\mathbf{r}_{ij}\|$  to ensure that the contributions  $E_i$  are local and no discontinuities are introduced when atoms enter or leave the cutoff radius. Even though such a decomposition is inherently nonunique and no labels for atom-wise energies are available in the reference data, they can still be approximated by a Gaussian process. The sum over atomic environments can be moved into the kernel function, yielding a kernel for systems  $\mathbf{x}$  and  $\mathbf{x}'$  with  $N$  and  $N'$  atoms, respectively:

$$K(\mathbf{x}, \mathbf{x}') = \sum_{i=1}^N \sum_{j=1}^{N'} K_{\text{local}}(x_i, x'_j) \quad (20)$$

Thus, reference energies for the whole system are sufficient for the model to learn a suitable energy decomposition into atomic environments.

Several descriptors and kernels for GAPs have been developed based on a local “atomic density”  $\rho(\mathbf{r}) = \sum_i \delta(\mathbf{r} - \mathbf{r}_i)$ . Initially, Bartók et al.<sup>179</sup> proposed to employ local atomic coordinates projected onto a 4D hyper sphere. Since this projection can represent the volume of a 3D sphere, the introduction of an additional radial basis can be avoided. To achieve rotational invariance, the bispectrum of 4D spherical harmonics of these coordinates was used as a descriptor. Alternatively, the SOAP (smooth overlap of atomic positions) kernel<sup>117</sup> is defined as the integral over rotations  $\mathcal{R}$  of atomic densities

$$K(\rho, \rho') = \int d\mathbf{R} \left| \int \rho(\mathbf{r}) \rho'(\mathcal{R}\mathbf{r}) d\mathbf{r} \right|^n \quad (21)$$

Given smoothed local densities  $\rho(\mathbf{r}) = \sum_i \exp(-\gamma\|\mathbf{r} - \mathbf{r}_i\|^2)$ , it has been shown that the SOAP kernel is equivalent to the linear kernel over the  $\text{SO}(3)$  power spectrum and bispectrum for  $n = 2$  and  $n = 3$ , respectively.<sup>117</sup> Both approaches are invariant to permutation of neighboring atoms as well as the rotation of the local environment. Further representations include best matches of the atomic densities over rotations<sup>177</sup> and kernels for symmetry-adapted prediction of tensorial properties.<sup>181,182</sup>

**2.3.4. Neural Network Potentials.** The first neural network potentials (NNPs) used a set of internal coordinates, for example, distances and angles, as structural representation to model the PES.<sup>183–187</sup> While being rototranslationally invariant, internal coordinates impose an arbitrary order on the atoms and are thus not reflecting the equivalence of permuted inputs. As a result, the NNP might assign different energies to symmetrically equivalent structures. Beyond that, the number of atoms influences the dimensionality of the input  $\mathbf{x}$ , limiting the applicability of the PES to chemical systems of the same size. Decomposing the energy prediction in the spirit of a many-body expansion circumvents these issues,<sup>188–190</sup> however, it scales unfavorably with system size and number of chemical species, because each term in the many-body expansion has to be modeled by a separate NN.

Behler and Parrinello<sup>114</sup> were the first to propose so-called high-dimensional neural networks potentials (HDNNPs), where the total energy of a chemical system is expressed as a sum over atomic contributions  $E = \sum_i E_i$ , predicted by the same NN (or one for each element). The underlying assumption is that the energetic contribution  $E_i$  of each atom depends mainly on its local chemical environment. As all atoms of the same type are treated identically and summation is

commutative, the output does not change when the input is permuted. Because of the decomposition into atomic contributions, systems with varying numbers of atoms can be predicted by the same NNP. In principle, this framework also enables transferability between system sizes, for example, a model can be trained on small systems, but applied to predict energies and forces for larger systems. However, this requires sufficient sampling of the local environments to remove spurious correlations caused by the training data distribution, as well as corrections for long-range effects.

The introduction of HDNNPs inspired many NN architectures that can be broadly categorized into two types. Descriptor-based NNPs<sup>116,191–193</sup> rely on fixed rules to encode the environment of an atom in a vector  $\mathbf{x}$ , which is then used as input for an ordinary feed-forward NN (see eq 16). These architectures include many variants of the original Behler-Parrinello network, such as ANI<sup>194</sup> and TensorMol.<sup>195</sup> On the other hand, end-to-end NNPs<sup>160,196–198</sup> take nuclear charges and Cartesian coordinates as input and learn a suitable representation from the data.

Many end-to-end NNPs have been inspired by the graph neural network by Scarselli et al.<sup>199</sup> and were later collectively cast as message-passing neural networks (MPNNs).<sup>198</sup> In this type of model, molecules are regarded as undirected graphs, where atoms are represented by nodes and interactions between them as edges. By exchanging information between nodes along edges (message-passing), complex chemical interactions can be modeled. A prominent example is the Deep Tensor Neural Network (DTNN).<sup>160</sup> Since its introduction, this approach has been refined to create new architectures, such as SchNet,<sup>109,200</sup> HIP-NN,<sup>115</sup> or PhysNet.<sup>108</sup> End-to-end NNPs that do not directly fall into the category of MPNNs are covariant compositional networks that are able to employ features of higher angular momentum<sup>201–203</sup> as well as models using a pseudodensity as input.<sup>172</sup>

Because no fixed rule is used to construct descriptors, end-to-end NNPs are able to automatically adapt the environment representations  $\mathbf{x}$  to the reference data (in contrast to the descriptor-based variant). However, as long as  $\mathbf{x}$  is invariant with respect to translation, rotation, and permutation of symmetry equivalent atoms, both types of NNPs adhere to all physical constraints outlined in Section 2.3.1. NNPs are commonly used to predict energies, while conservative forces are obtained by derivation. Despite being energy-based, it is still possible to incorporate information from *ab initio* forces by including them in the loss term optimized during training. At this point, it should be noted that the requirement for continuously differentiable models excludes the use of certain activation functions, for example the popular ReLU activation,<sup>204</sup> when constructing ML-FFs based on neural networks. To avoid discontinuities in the forces, activation functions used for NNPs must always be smooth.

**2.3.4.1. Descriptor-Based NNPs.** The first descriptor-based NNP introduced by Behler and Parrinello<sup>114</sup> uses atom-centered symmetry functions (ACSFs)<sup>116</sup> consisting of two-body terms

$$G_i^2 = \sum_{j \neq i} e^{-\eta(r_{ij}-r_s)^2} f_{\text{cut}}(r_{ij}) \quad (22)$$

and three-body terms

$$G_i^3 = 2^{1-\zeta} \sum_{j,k \neq i} (1 + \lambda \cos \theta_{ijk})^\zeta e^{-\eta(r_{ij}^2 + r_{ik}^2 + r_{jk}^2)} \times f_{\text{cut}}(r_{ij}) f_{\text{cut}}(r_{ik}) f_{\text{cut}}(r_{jk}) \quad (23)$$

to encode information about the chemical environment of each atom  $i$ . Here,  $r_{ij}$  is the distance between atoms  $i$  and  $j$ ,  $\theta_{ijk}$  the angle spanned by atoms  $i$ ,  $j$  and  $k$  centered around  $i$ , and the summations run over all atoms within a cutoff distance  $r_{\text{cut}}$ . As the atom order is irrelevant for the values of  $G_i^2$  and  $G_i^3$  and only internal coordinates are used to calculate them, all physical invariants are satisfied. A cutoff function such as

$$f_{\text{cut}}(r) = \begin{cases} \frac{1}{2} \left[ \cos \left( \frac{\pi r}{r_{\text{cut}}} \right) + 1 \right], & r \leq r_{\text{cut}} \\ 0, & r > r_{\text{cut}} \end{cases} \quad (24)$$

ensures that  $G_i^2$  and  $G_i^3$  vary smoothly when atoms enter or leave the cutoff sphere and the parameters  $\eta$ ,  $r_{\text{cut}}$ ,  $\zeta$ , and  $\lambda (= \pm 1)$  determine to which distances, or combinations of angles and distances, the ACSFs are most sensitive. When sufficiently many  $G_i^2$  and  $G_i^3$  with different parameters are combined and stored in a vector  $\mathbf{x}_i$ , they form a “fingerprint” of the local environment of atom  $i$ . This environment descriptor is then used as input for a neural network for predicting the energy contributions  $E_i$  of atoms  $i$  and the total energy  $E = \sum_i E_i$  is obtained by summation.

Since the ACSFs only use geometric information, they work best for systems containing only atoms of one element, for example crystalline silicon.<sup>114</sup> To describe multicomponent systems, typically, the symmetry functions are duplicated for each combination of elements and separate NNs are used to predict the energy contributions for atoms of the same type.<sup>205</sup> Since the combinatorial explosion can lead to a large number of ACSFs for systems containing many different elements, an alternative is to modify the ACSFs with element-dependent weighting functions.<sup>206</sup> Most descriptor-based NNPs, such as ANI<sup>194</sup> and TensorMol,<sup>195</sup> use variations of eqs 22 and 23 (sometimes allowing parameters of ACSFs to be optimized during training) to construct the environment descriptors  $\mathbf{x}_i$ . Different ways to encode the structural information are possible, for example using three-dimensional Zernike functions,<sup>191</sup> or the coefficients of a spherical harmonics expansion,<sup>193</sup> but the general principle remains the same. Also, while most descriptor-based NNPs use separate parametrizations for different elements, it is also possible to use a single NN to predict all atomic energy contributions.<sup>193</sup> The common feature for all variations of this approach is that the functional form of the environment descriptor is predetermined and manually designed.

**2.3.4.2. End-to-End NNs.** A potential drawback of the previously introduced ACSFs is that they must be chosen by an expert before training the neural network. If the choice of symmetry functions is poor, for example when the resulting descriptor is (nearly) identical for two very different structures, the expressive power of the neural network and the achievable accuracy are limited *a priori*. Additionally, a growing number of input dimensions can quickly become computationally expensive, both for calculating the descriptors and for evaluating the NN. This is especially the case when modeling multicomponent systems, where commonly orthogonality is assumed between different elements (which increases the

number of input dimensions) or the descriptors are simply weighted by an element-dependent factor (which may limit the structural resolution of the descriptor).

In contrast, end-to-end NNPs directly take atomic types and positions as inputs to learn suitable representations from the reference data. Similar to descriptor-based NNPs, many end-to-end NNPs obtain the total energy  $E$  as a sum of atomic contributions  $E_i$ . However, those are predicted from learned features  $\mathbf{x}_i$  encoding information about the local chemical environment of each atom  $i$ . This allows them to adapt the features based on the size and distribution of the training set as well as the chemical property of interest during the training process. The idea is to learn a mapping to a high-dimensional feature space, so that structurally (and energetically) similar atomic environments lie close together and dissimilar ones far apart.

Within the deep tensor neural network framework,<sup>160</sup> this is achieved by iteratively refining the atomic features  $\mathbf{x}_i$  based on neighboring atoms. The features are initialized to  $\mathbf{x}_i^2 = \mathbf{e}_Z$ , where  $\mathbf{e}_Z$  are learnable element-dependent representations that are updated for  $T \in [3, 6]$  steps. This procedure is inspired by diffusion graph kernels<sup>207</sup> as well as the graph neural network model by Scarselli et al.<sup>199</sup> Many end-to-end networks have adapted this approach which can be written in general as

$$\mathbf{x}_i^{t+1} = \mathcal{G}^t \left( \mathbf{x}_i^t + \sum_{j \neq i} \mathcal{F}^t(\mathbf{x}_i^t, \mathbf{x}_j^t, \mathbf{g}(r_{ij})) f_{\text{cut}}(r_{ij}) \right) \quad (25)$$

where the summation runs over all atoms within a distance  $r_{\text{cut}}$  and a cutoff function  $f_{\text{cut}}$  ensures smooth behavior when atoms cross the cutoff. Here, the “atom-wise” function  $\mathcal{G}^t$  is used to refine the atomic features after they have been updated with information from neighboring atoms through the interaction-function  $\mathcal{F}^t$ . Usually, the interatomic distance  $r_{ij}$  is not used directly as input to  $\mathcal{F}^t$ , but expanded in a set of uniformly spaced radial basis functions<sup>108,109,160</sup> to form a vectorial input  $\mathbf{g}(r_{ij})$ . Both  $\mathcal{F}^t$  and  $\mathcal{G}^t$  functions are NNs with the specific implementations varying between different end-to-end NNP architectures. As only pairwise distances are used and the order of atoms is irrelevant due to the commutative property of summation, the features  $\mathbf{x}_i$  obtained by eq 25 are automatically rototranslationally and permutationally invariant (and thus also the energy predictions).

Gilmer et al.<sup>198</sup> have cast graph networks of this structure as message-passing neural networks and proposed a variant that uses a set2set decoder<sup>208</sup> instead of a sum over energy contributions to achieve permutational invariance of the energy. SchNet<sup>109</sup> takes an alternative view of the problem and models interactions between atoms with convolutions. The convolution filters need to be continuous (to have smooth predictions) but are evaluated at finite points, i.e., the positions of neighboring atoms. To ensure rotational invariance, only radial convolution filters are used, leading again to an interaction function that is a special case of eq 25.

While the previously introduced approaches aim to learn as much as possible from the reference data, several models have been proposed to better exploit chemical domain knowledge. The hierarchical interacting particle neural network (HIP-NN)<sup>115</sup> obtains the prediction as a sum over atom-wise contributions  $E_i^t$  that are predicted after every update step  $t$ . A regularizer penalizes larger energy contributions in deeper layers, enforcing a declining, hierarchical prediction of the

energy. PhysNet<sup>108</sup> modified the energy function to include explicit terms for electrostatic and dispersion interactions

$$E = \sum_{i=1}^N E_i + k_e \sum_{i=1}^N \sum_{j>i}^N \tilde{q}_i \tilde{q}_j \chi(r_{ij}) + E_{D3} \quad (26)$$

where  $E_{D3}$  is Grimme's D3 dispersion correction,<sup>209</sup>  $k_e$  is Coulomb's constant, and  $\tilde{q}_i$  are corrected partial charges predicted by the network that are guaranteed to sum to the total charge of the molecule. In an ablation study on a data set of  $S_N2$  reactions,<sup>210</sup> it was shown that the inclusion of long-range terms improves prediction accuracy for energies and forces while models without these terms show qualitatively wrong asymptotic behavior.<sup>108</sup>

### 3. BEST PRACTICES AND PITFALLS

A number of careful modeling steps are necessary to construct an ML-FF for a particular problem of interest (Figure 13). Even before starting this process, some forethought is appropriate due to certain limitations of *ab initio* methods themselves. This section gives an overview about all steps necessary to construct an ML-FF from scratch and highlights possible "pitfalls", that is, issues that may occur along the way, in particular when the recommended practices are not followed. First, some preliminary considerations, which should be taken before starting with the construction of an ML-FF, are discussed (Section 3.1). Next, basic principles for choosing an appropriate ML method for a specific task are given (Section 3.2). Then the importance of high quality reference data, different strategies to collect it (Section 3.3), and how the data has to be prepared (Section 3.4) are outlined. This is followed by an overview of how to train an ML model on the collected data (Section 3.5) and guidelines for using the trained ML-FF in a production setting, for example, for running MD simulations (Section 3.6). Finally, popular software packages for constructing ML-FFs are briefly described and code examples are given (Section 3.7).

#### 3.1. Preliminary Considerations

Before running any *ab initio* calculations to collect data for training ML models, it is advisable to think about the limitations of the chosen level of theory itself. The issues discussed here are problem-specific and often not unique to ML-FFs, but PES reconstruction in general. As such, a comprehensive list is not possible, but a few examples are given below.

**3.1.1. Practicability.** On the spectrum of quantum chemistry methods, ML-FFs fit into the niche between highly efficient conventional FFs<sup>211</sup> and accurate, but computationally expensive *ab initio* methods.<sup>212</sup> Efficiency-wise, they are still inferior to classical FFs, because their functional forms are considerably more complex and thus more expensive to evaluate. Even the fastest ML-FFs are still one to three orders of magnitude slower.<sup>165,213,214</sup> On the other end, ML-FFs are lower bounded by the accuracy of the reference data used for training, which means that the underlying *ab initio* method will always be at least equally accurate. In practical terms, this means that to be useful, ML-FFs need to offer time savings over directly running *ab initio* calculations and an improved accuracy compared to conventional FFs. For this purpose, the full procedure of data generation, training and inference must be taken into account, as opposed to just regarding inference speed, which will be much quicker than *ab initio* methods.

While this consideration sounds trivial at first, it is still advisable to think about whether constructing an ML-FF really is economical. For example, if the goal is to run just a single short MD trajectory, the question is how much data is necessary for the model to reach the required accuracy. Some models may require several thousands of training points to produce accurate enough predictions, even for fairly small molecules. Then when factoring in the overall time required for going through the process of creating the ML model, testing it, and running the MD simulation, it might be more efficient to simply run an *ab initio* MD simulation in the first place. Further, not every ML method is equally applicable or appropriate for all systems due to methodical or conceptual constraints. Such limitations are discussed in greater detail in Section 3.2.

**3.1.2. Multireference Effects.** Many *ab initio* methods use a single Slater determinant to express the wave function of a system. The problem with this approach is that different determinants may be dominant in different regions of the PES, leading to a poor description of the wave function if the wrong determinant is chosen. Especially when many calculations are performed for various strongly distorted geometries, for example when a reaction is studied and bonds need to be broken, it may happen that the solution "jumps" discontinuously from one electronic state to another, leading to inconsistent reference data. When an ML model is trained on such a data set, it will try to find a compromise between the inconsistencies and its performance typically be unsatisfying. It is therefore advisable to check for possible multireference effects prior to generating data and, if necessary, switch to a multireference method (for a comprehensive review on multireference methods, see ref 215).

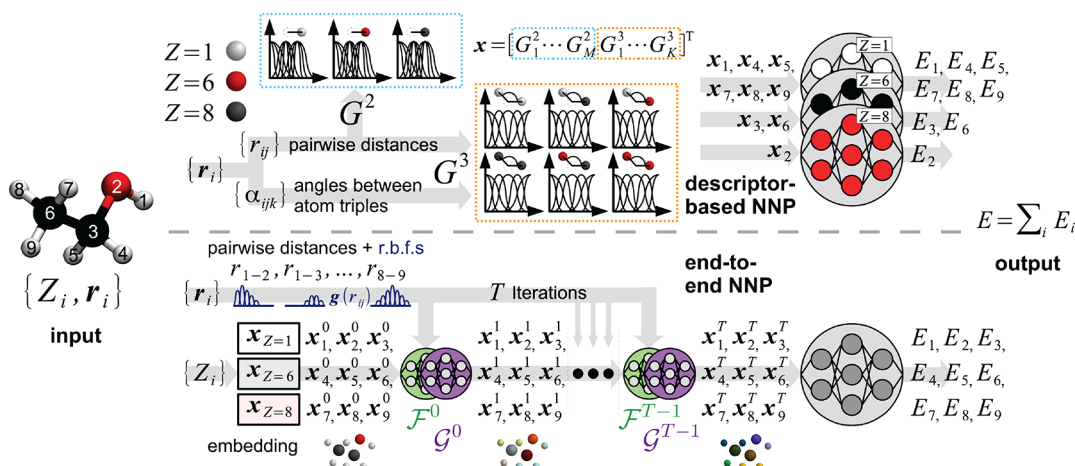
**3.1.3. Strong Delocalization.** The models discussed in this review all assume that energy contributions are local to some degree. This assumption is either introduced explicitly by a cutoff radius, or it enters the model through the use of a specific structural descriptor. For example, by using inverse distances to encode chemical structures for kernel methods (as is done, e.g., in GDML, see Section 2.3.2), relative changes between close atoms are weighed more strongly when comparing two conformations. While assuming locality is valid in many practical applications, there exist many cases where this assumption breaks down. An example are extensive conjugated  $\pi$ -systems, where a rotation around certain bonds might break the favorable interaction between electrons, leading to a "non-local" energy contribution. If such effects exist, an appropriate model should be chosen, for example the cutoff radius may need to be larger than usual or a different structural descriptor must be picked.

#### 3.2. Choosing an Appropriate ML Method

Several different variants of ML-FFs have been discussed in Section 2.3, and many more are described in the literature. Although all these methods can be applied to construct ML-FFs for any chemical system, some methods might be more promising than others for certain tasks. For researchers who want to apply ML methods to a specific problem for the first time, the abundance of different models to choose from may be overwhelming and it might be difficult to find an appropriate choice.

In the following, possible applications of ML-FFs are broadly categorized based on simple questions about the task at hand. For each case, advantages and disadvantages of individual





**Figure 12.** Overview of descriptor-based (top) and end-to-end (bottom) NNPs. Both types of architecture take as input a set of  $N$  nuclear charges  $Z_i$  and Cartesian coordinates  $\mathbf{r}_i$  and output atomic energy contributions  $E_i$ , which are summed to the total energy prediction  $E$  (here  $N = 9$ , an ethanol molecule is used as example). In the descriptor-based variant, pairwise distances  $r_{ij}$  and angles  $\alpha_{ijk}$  between triplets of atoms are calculated from the Cartesian coordinates and used to compute hand-crafted two-body ( $G^2$ ) and three-body ( $G^3$ ) atom-centered symmetry functions (ACSFs) (see eqs 22 and 23). For each atom  $i$ , the values of  $M$  different  $G^2$  and  $K$  different  $G^3$  ACSFs are collected in a vector  $\mathbf{x}_i$ , which serves as a fingerprint of the atomic environment and is used as input to an NN predicting  $E_i$ . Information about the nuclear charges is encoded by having separate NNs and sets of ACSFs for all (combinations of) elements. In end-to-end NNPs,  $Z_i$  is used to initialize the vector representation  $\mathbf{x}_i^0$  of each atom to an element-dependent (learnable) embedding (atoms with the same  $Z_i$  start from the same representation). Geometric information is encoded by iteratively passing these descriptors (along with pairwise distances  $r_{ij}$  expanded in radial basis functions  $g(r_{ij})$ ) in  $T$  steps through NNs representing interaction functions  $\mathcal{F}^t$  and atom-wise refinements  $\mathcal{G}^t$  (see eq 25). The final descriptors  $\mathbf{x}_i^T$  are used as input for an additional NN predicting the atomic energy contributions (typically, a single NN is shared among all elements).

models are discussed to provide help and guidance to the reader for identifying an appropriate model for their use case.

**How much reference data can be used for training?** *When in doubt which method to use, a rule of thumb could be to prefer kernel methods when there are less than  $\sim 10^3$ – $10^4$  training points and NN-based approaches otherwise (but this may also be a matter of preference).*

Depending on the desired accuracy, the amount of *ab initio* reference data which can be collected within a reasonable time frame may be vastly different. For example, if reference calculations are performed at the DFT level of theory, it is often feasible to collect several thousands of data points, even for relatively large molecules. On the other hand, if CCSD(T) accuracy and a large basis set is required, already a few hundred reference calculations for small molecules can require a considerable amount of computing time. Although it is of course always desirable to perform as few reference calculations as possible, for some tasks, collecting a large data set is unavoidable. For example, if a model should be able to predict a variety of different molecules containing many different elements, the relevant chemical space must be sampled sufficiently.

In general, kernel-based models tend to achieve good prediction accuracies even with few training points, whereas NNs often need more data to reach their full potential (although there may be exceptions for both model variants, see also Figure 5). Further, the optimal model parameters for kernel models can be determined analytically (see eq 10), which, at least for small data sets, is typically faster than training a NN via (a variant of) stochastic gradient descent. However, when the data set size  $M$  is very large, solving eq 10 analytically can become prohibitively expensive as it scales  $O(M^3)$  (and requires  $O(M^2)$  memory to store the kernel matrix). Further, evaluating kernel models scales with  $O(M)$  (see eq 2), whereas the cost of evaluating NN-based methods

has (as long as the number of parameters does not have to be increased for larger data sets) constant complexity. For this reason, NNs tend to be more suitable for large data sets. Note that there are approximations which improve the scaling of kernel methods (so they can be applied even to very large data sets) at the cost of accuracy (see eqs 11–13).

**Should the model be able to predict a single type of chemical system or multiple different ones?** *To be applicable to multiple systems, a model must decompose its prediction into atomic contributions. Models that use no such decomposition must either use a fixed size descriptor or several separate models need to be trained.*

Some ML-FFs only need to be able to predict systems with a fixed composition and number of atoms, for example to study the dynamics of a single molecule, whereas other applications require the ability to predict different systems with varying size, for example, when clusters consisting of a different number and kind of molecules are studied with the same model.

While all ML-FFs can be applied in the first case, the latter requires either that the length of chemical descriptors is independent of the number of atoms, or that model predictions are decomposed into local contributions based on fixed-size fingerprints of atomic environments (which naturally makes them extensive). Most NNPs (see Section 2.2.2) and many kernel methods, for example, GAPs (see Section 2.3.3) or FCHL,<sup>106,107</sup> use such a decomposition and can be applied to differently sized chemical systems without issues. Exceptions are, for example, (s)GDML models (see Section 2.3.2), which encode chemical structures as vectors of inverse distances between atomic pairs. Consequently, the length of the descriptor changes with the number of atoms and the model can only be applied to a single type of system. In some special cases, it may be possible to choose a maximum descriptor length and pad descriptors of smaller molecules with

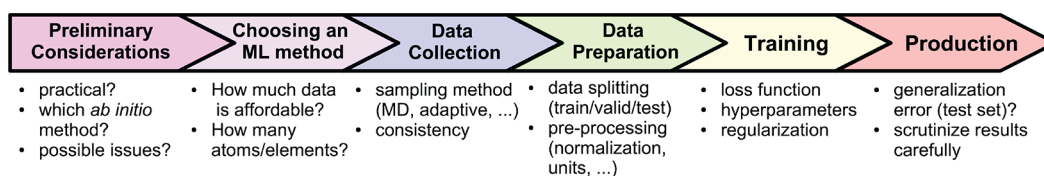


Figure 13. Overview of the most important steps when constructing and using ML-FFs.

zeros, but this may introduce other problems or reduce the accuracy.

**Will the model be applied to single or multicomponent systems?** If only a handful of elements is relevant, all models are equally suitable. When a large number of elements needs to be considered, the model should be able to encode and use information about atom types efficiently.

As long as an ML-FF is only applied to single-component systems (consisting of a single element), for example elemental carbon or silicon, all relevant information is contained in the relative arrangement of atoms and nuclear charges need not be encoded explicitly. However, as soon as there are multiple atom types (as is common for most applications of ML-FFs), the model must have some way to distinguish between them. A notable exception are some models such as (s)GDML, which use inverse pairwise distances as structural descriptor. Here, information about atom types is implicitly contained, because specific entries always correspond to the same combination of atom types.

Many local descriptors of atomic environments only use geometric information in the form of distances and angles between pairs and triplets of atoms (see eqs 22 and 23). To include information about atomic types, geometric features have to be included separately for every possible combination of elements, leading to a drastic increase of descriptor size (descriptors for kernel machines based, for example, on SOAP<sup>117</sup> or FCHL<sup>106,107</sup> also grow in size when the number of atom types is increased). Many descriptor-based NNPs further use separate NNs to predict atomic contributions of different elements (see Figure 12). A disadvantage of these approaches is that the number of terms in the descriptor increases combinatorially with the number of elements covered by the model (in particular if three-body or even four-body terms are used), which impacts the computational cost of training and evaluating the model. Also, larger amounts of training data may become necessary for good results. As long as only a few elements need to be considered, these downsides are not an issue, but if a model for a significant fraction of the periodic table is required, a more efficient representation is desirable. Most end-to-end NNPs employ so-called element embeddings (see Figure 12), which do not become more complex when the number of elements is increased. This has the additional benefit of potentially increasing the data efficiency of the model by utilizing alchemical information. Another alternative is to introduce element-dependent weighting functions (instead of duplicating terms in ACSF descriptors).<sup>206</sup>

**Will the model be applied to small or large systems?** Models for very large target systems should be able to exploit chemical locality, so that reference calculations for fragments can be used as training data. Additionally, this allows trivial parallelization of predictions over multiple machines.

Often, ML-FFs are used to study small or medium-sized molecules. In such cases, all models are equally applicable. For

very large systems containing many atoms, however, some methods have particularly advantageous properties. For example, it might be infeasible to run *ab initio* calculations for the full target system. In this case, being able to fragment the system into smaller parts, for which reference calculations are affordable, is very useful.

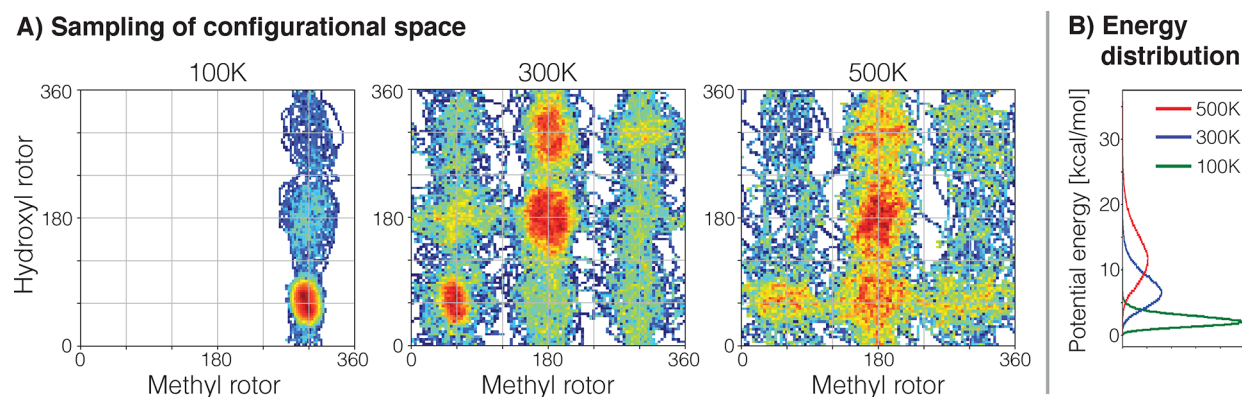
To be trainable on such fragments, ML-FFs must introduce an explicit assumption about chemical locality by introducing a cutoff radius. Every method that decomposes predictions into a sum of local atomic contributions can thus be trained in this way. ML-FFs without cutoffs on the other hand need reference data for the complete system (see above). Another advantage of local models is that their predictions are embarrassingly parallel. The contributions of individual atoms can be calculated on separate machines (storing a copy of the model), each requiring only information about neighboring atoms within the cutoff radius. Apart from possible efficiency benefits, this may even become necessary if the computations to handle all atoms do not fit into the memory of a single machine (for example when the system of interest consists of millions of atoms<sup>216</sup>). Note that while not all ML methods to construct FFs can be parallelized in this way, most models contain mostly linear operations, which are amenable to other parallelization methods, for example, by utilizing GPUs (graphics processing units).

At this point, a subtle difference between cutoffs used in NNPs of the message-passing type (see Section 2.3.4) and descriptor-based NNPs (as well as kernel machines based on local atomic environments) should be pointed out. In message-passing schemes, information between all atoms within the cutoff radius is exchanged over  $T$  iterations, thus the effective cutoff radius increases by a factor of  $T$ . This means that to distribute the computation over multiple machines, it is either necessary to communicate updates to other machines after each iteration, or a sufficiently large subdomain needs to be stored on all machines.

Are long-range interactions expected to be important for the system of interest? If strong long-range contributions to the energy are present, it is advisable to either use a model without cutoffs, or augment the pure ML approach by explicitly including physical interaction terms.

As described earlier, many ML-FFs introduce cutoffs to exploit chemical locality. An obvious downside of this approach is that all interactions beyond the cutoff cannot be represented. For uncharged molecules without strong dipole moments, relevant interactions are usually sufficiently short-ranged that this is not problematic. However, when strong long-ranged (e.g., charge-dipole) interactions are important, cutoffs may introduce significant errors. Models such as (s)GDML, which consider the whole chemical structure without introducing cutoffs, do not suffer from this issue in principle.

While it is possible to simply increase the cutoff distance until more long-ranged contributions can be neglected, this



**Figure 14.** (A) Two-dimensional projection of the sampled regions of the PES of ethanol at 100 K, 300 K, and 500 K from running AIMD simulations with FHI-aims<sup>218</sup> (Fritz Haber Institute *ab initio* molecular simulations) at the PBE+TS/DFT level of theory.<sup>167,168</sup> The length of the simulation was 500 ps. (B) Distribution of sampled potential energies for the three different temperatures.

decreases the computational and data efficiency of models which were designed with cutoffs in mind. A better alternative could be to include the relevant physical interaction terms explicitly in the model. For example, TensorMol<sup>195</sup> and PhysNet<sup>108</sup> include such correction terms by default, but other models can be augmented in a similar fashion. Although not strictly necessary, even models without cutoffs may profit from such terms by an increased data efficiency.

### 3.3. Data Collection

A fundamental component of any ML model is the reference data. While its architecture and other technical details are responsible for the potential accuracy of a model, the choice of reference data and its quality defines the reliability and range of applicability of the final model. Any deficiencies that are present in the data will inevitably also lead to artifacts in models trained on it, a principle often colloquially stated as “garbage in, garbage out”.<sup>217</sup> As such, the reference data is one of the most important components of an ML-FF. The generation of data sets in computational chemistry and physics are challenges on their own. First of all, each reference point is a result of computationally expensive and often nontrivial calculations (see Section 2.1), which limits the amount of data that can be collected. Furthermore, the dimensionality of the configurational space of molecules, solids, or liquids is so vast that—except for trivial cases—it is not apparent how to identify the representative geometries in the ocean of possibilities. The optimal choice of reference data might even need to be adapted to the individual properties of the respective ML model that consumes it or its intended application. In the following, several strategies for sampling the PES and generating reference data sets are outlined (multiple of these approaches can be combined). Afterward, problems that may occur due to insufficient sampling are highlighted and general remarks about the importance of a consistent reference data set are given.

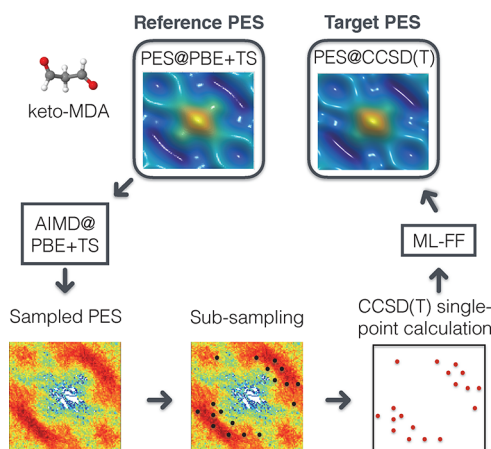
**3.3.1. AIMD Sampling.** A good starting point to assemble the reference data set is by sampling the PES using *ab initio* molecular dynamics (AIMD) simulations. Albeit expensive in terms of the amount of necessary reference calculations, this technique constitutes a straightforward way to explore configurational space. Here, the temperature of the simulation determines which regions of the PES and what energy ranges (according to the Boltzmann distribution) are explored (see Figure 14). For example, if the aim is to construct an ML-FF

for calculating the vibrational spectrum of ethanol at 300 K, generating the database at 500 K is a safe option since the subspace of configurations relevant at 300 K is contained in the resulting database (see Figure 14A). Sampling at higher temperatures ensures that the model does not enter the extrapolation regime during production runs, which is practically guaranteed to happen when a lower temperature is used for sampling. In general, the resulting data set will be biased toward lower energy regions of the PES, where the system spends most of the simulation time. For this reason, pure AIMD sampling is only advisable when the intended application of the final ML model involves MD simulations for equilibrium or close to equilibrium properties, where rare events do not play a major role. Examples of this are the study of vibrational spectra, minima population, or thermodynamic properties.

**3.3.2. Sampling by Proxy.** Constructing reliable reference data sets from AIMD simulations can be computationally expensive. While system size plays a major role, other phenomena, such as the presence of intramolecular interactions and fluxional groups, can also influence how quickly the PES is explored. Because of this, long simulation times may be required to visit all relevant regions. For example, generating  $2 \times 10^5$  conformations from AIMD using a relatively affordable level of theory (e.g., PBE+TS/DFT with a small basis set) can take between a few days to several weeks (depending on the size of the molecule). With higher levels of theory, the required computation time may increase to months, or, when highly accurate methods such as CCSD(T) are required, even become prohibitively long (several years).

To resolve this issue, a possible strategy is to sample the PES at a lower level of theory to generate a long trajectory that covers many regions on the PES. The collected data set is then subsampled to generate a small, but representative set of geometries, which serve as input for performing single-point calculations at a higher level of theory (see Figure 15). This strategy works best when the PES has a similar topology at both levels of theory, so it can be expected that configurations generated at the lower level are representative of configurations that would be visited in an AIMD simulation at the higher level (see the two-dimensional projections of the PES in Figure 15). When the two PESs are topologically very different, for example, when a semiempirical method or even a conventional FF is used to generate the initial trajectory, it may happen that the relevant regions of the PES at the higher level of theory are





**Figure 15.** Procedure followed to generate a database at the CCSD(T) level of theory for keto-malondialdehyde using sampling by proxy. An AIMD simulation at 500 K computed at the PBE+TS/DFT level of theory is used to sample the molecular PES. Afterward, the trajectory is subsampled (black dots) to generate a subset of representative geometries, for which single-point calculations at the CCSD(T) level of theory are performed (red dots). This highly accurate reference data is then used to train an ML-FF.

not covered sufficiently. Then, when an ML-FF is trained on the collected data set and used for running an MD simulation, the trajectory may enter the extrapolation regime and the model might give unphysical predictions. Thus, extra care should be taken when two very different levels of theory are used for sampling by proxy.

**3.3.3. Adaptive Sampling.** Another method to minimize the amount of expensive *ab initio* calculations is called adaptive sampling or on-the-fly ML.<sup>219</sup> Here, a preliminary ML-FF is trained on only a small initial set of reference data and then used to run an MD simulation. During the dynamics, additional conformations are collected whenever the model predictions become unreliable according to an uncertainty criterion. Then, new reference calculations are performed for the collected structures and the training of the ML model is continued or started from scratch on the augmented data set. The process is repeated until no further unreliable regions can be discovered during MD simulations.

When following this strategy, the quality of the uncertainty estimate is crucial for an efficient sampling of the PES: If the estimate is overconfident, deviations from the reference PES might be missed. If the estimate is overly cautious, many redundant *ab initio* calculations have to be performed. There exist several ways to estimate the uncertainty of an ML-FF. For example, Bayesian methods learn a probability distribution over models, which enables straightforward uncertainty estimates (see the predictive variance of a Gaussian process, eq 9). For models where an explicit uncertainty estimate is not available, for example, neural networks, a viable alternative is query-by-committee.<sup>205,220</sup> Here, an ensemble of models is trained, for example on different subsets of the reference data and each starting from a different parameter initialization. Then, the discrepancy between their predictions can be used as uncertainty estimate. Query-by-committee has been successfully employed to sample PESs using neural networks for water dimers,<sup>221</sup> organic molecules<sup>71,108</sup> as well as across chemical compound space.<sup>222</sup> Other alternatives, for example using dropout<sup>223</sup> as a Bayesian approximation,<sup>224</sup> could also be used.

Collecting data “on-the-fly” is even possible without uncertainty estimates. Instead, additional reference calculations are performed at fixed intervals during the MD simulation.<sup>219,225</sup> This relies on the assumption that the probability of reaching the extrapolation regime of an ML model rises with increasing length of the MD trajectory. While performing *ab initio* calculations in regular intervals will discover all deviations of the model eventually, this variant of on-the-fly ML does not exploit any information about the already collected reference set and may thus lead to many redundant data points. More detailed reviews on uncertainty estimation and active sampling of PESs can be found in refs 226 and 227.

**3.3.4. Metadynamics Sampling.** Similar to adaptive sampling, metadynamics sampling<sup>228,229</sup> uses a preliminary ML-FF to run MD simulations to find structures for which to run reference calculations. However, the dynamics are biased to increase the probability for visiting unexplored regions on the PES. This is achieved by placing “Gaussian bump functions” on the PES in regions that have already been visited, raising the potential energy of already known structures artificially. It is possible to combine metadynamics with the uncertainty estimates used in adaptive sampling to only select the most relevant structures.

**3.3.5. Normal Mode Sampling.** It is also possible to sample the PES without running any kind of MD simulation. In normal mode sampling,<sup>194</sup> the idea is to start from a minimum on the PES and generate distorted structures by randomly displacing atoms along the normal modes. They are the eigenvectors of the mass-weighted Hessian matrix obtained at the minimum position, that is, a harmonic approximation of the molecular vibrations. From the associated force constants (related to the eigenvalues), the increase in potential energy for displacements along individual normal modes can be estimated. Since they are orthogonal to each other, it is straightforward to combine multiple random displacements along different normal modes such that the resulting structures are sampled from a Boltzmann distribution at a certain temperature. In other words, structures generated like this are drawn from the same distribution as if an “approximated PES” was sampled with a (sufficiently long) MD simulation. This approximated PES is equivalent to a Taylor expansion of the original PES around the minimum position, truncated after the quadratic term (the contribution of the linear term vanishes at extrema).

Structures generated from random normal mode sampling are not correlated, in contrast to those obtained from adjacent time steps in MD simulations, which makes this approach an efficient way to explore the PES. However, the disadvantage is that only regions close to minima can be sampled. Additionally, the harmonic approximation is only valid for small distortions, meaning the larger the temperature, the more the sampled distribution diverges from the Boltzmann distribution on the true PES. Because of these limitations, it is best to combine normal mode sampling with other sampling methods, for example to generate an initial reference data set, which is later expanded by adaptive sampling.

**3.3.6. Problems Due to Insufficient Sampling.** Because their extrapolation capabilities are limited, ML methods only give reliable predictions in regions where training data is present.<sup>230</sup> When generating reference data, it is therefore important that all regions of the PES that may be relevant for a later study are sampled sufficiently. For example, when studying a reaction, the data should not only cover

configurations corresponding to educt and product structures, but also the region around the transition state and along the transition pathway. When the reaction coordinate defining the transition process is already known, a straightforward way to generate the reference data would be to sample the transition path region. However, even when an ML model can reproduce the entire reference data set with the required accuracy, it is still possible to run into issues when the model is used to study the reaction. If the rare transition process was not sampled sufficiently, it is not guaranteed that MD simulations with the ML-FF reproduce it correctly. The reference data may be restricted to a specific subset of molecular configurations along the transition pathway. Hence, the model can enter the extrapolation regime somewhere between the boundary states and the transition pathways generated by an MD simulation might be unreasonable. Another potential issue is that after passing the transition state region, typically, a large amount of potential energy is converted to internal motions such as bond vibrations. As a result, the effective temperature defined by the kinetic energy exceeds the ambient conditions by orders of magnitude. Even when using a thermostat in the simulation, thermal energy increases so rapidly that it may not be able to handle the increase in temperature immediately. As a consequence, the trajectory visits high-energy configurations, which may not be included in the reference data, and the model again has to extrapolate.

When ML-FFs enter the extrapolation regime, i.e., they are used to predict values outside the sampled regions of the PES, unphysical effects may be observed. Consider for example the dissociation of the O–H bond in the hydroxyl group of ethanol (Figure 16). Here, different models were trained on data gathered from an MD simulation of ethanol at 500 K and used to predict how the energy changes when the O–H distance of the hydroxyl group is shortened or elongated to extreme values well outside the range sampled during the dynamics. In this example, while the sGDML model is able to

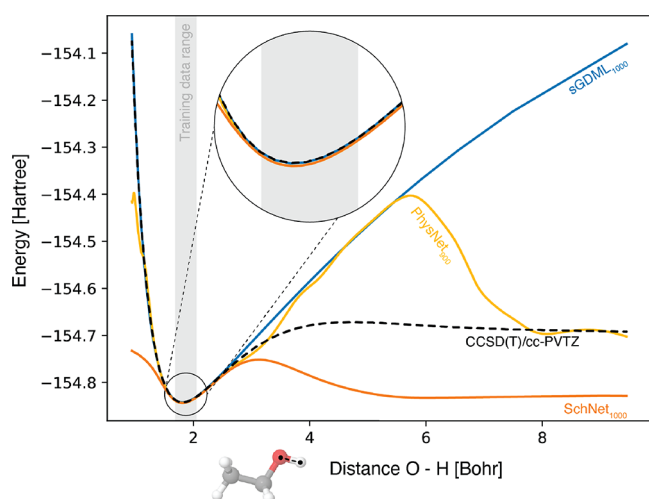
accurately extrapolate to much shorter distances than are present in the training data, it still fails to predict the bond dissociation. The NNP models (PhysNet and SchNet) exhibit qualitatively wrong short-range behavior and spurious minima on the PES, which may trap trajectories during MD simulations. Because of these limited extrapolation capabilities, it is advisable to sample larger regions of the PES than are expected to be visited during MD simulations so that there is a “buffer” and models never enter the unreliable extrapolation regime during production runs. For example, when an ML-FF is to be used for a study at a temperature of 300 K, the PES should be sampled around 500 K or higher.

**3.3.7. Importance of Data Consistency.** Although it may appear trivial, it is crucial that all data used for training a model is internally consistent: A single level of theory (method and basis set) should be used to calculate the reference data. When multiple quantum chemical codes (or even different versions of the same code) are used for data generation, it should be checked that their outputs are numerically identical when given the same input geometry (if they are not then this will effectively manifest itself like noisy outputs, severely deteriorating the precision of the ML model). Further, many *ab initio* codes automatically reorient the input geometry such that the principal moments of inertia are aligned with the *x*-, *y*-, and *z*-axes, so extra care should be taken when forces or other orientation-dependent quantities (i.e., electric moments) are extracted to verify they are consistent with the input geometry. When some calculation settings need to be adapted for a subset of the data, for example, for cases with difficult convergence, it is important to check that values computed with the modified settings are consistent with the rest of the data. Additionally, for training some ML models, it may be essential that atoms are ordered in a particular way throughout the data set. For example, the permutational symmetry of (s)GDML models is limited to the transformations recovered from the training set, whereas the NN models discussed in this review are fully agnostic with respect to atom indexing.

### 3.4. Data Preparation

After the reference data are collected, they must be prepared for the training procedure. This includes splitting the data into different subsets, which are reserved for separate purposes. Some models may also require that the data is preprocessed in some way before the training can start. In the following, important aspects of these preparation steps are highlighted.

**3.4.1. Splitting the Data.** Prior to training any ML model, it is necessary to split the reference data into disjoint subsets for training/validation and testing (see Section 2.2.3). While the training/validation set is used for fitting the model, the test set is only ever used after a model is trained to estimate its generalization error, i.e., to judge how well the model performs on unseen data.<sup>30,159</sup> It is very important to keep the two splits separate, as it is easily possible to achieve training errors that are several orders of magnitude lower than the true generalization error when the model is not properly regularized. Many models also feature hyperparameters, such as kernel widths, regularization terms or learning rates, that must be tuned by comparing several trained model variants on a third data set used purely for validation (a subset of the training/validation set). Note that information from the validation set will still enter the model indirectly, that is, it also participates in the training process. This is why a strict separation of the training/validation set from the test set is



**Figure 16.** One-dimensional cut through the PES of ethanol along the O–H bond distance for different ML-FFs (solid blue, yellow, and orange lines) compared to *ab initio* reference data (dashed black line). Close to the region sampled by the training data (range highlighted in gray), all model predictions are virtually identical to the reference method (see zoomed view). When extrapolating far from the sampled region, the different models have increasingly large prediction errors and behave unphysically.

crucial. Undetected duplicates in the data set can complicate splitting, as the contamination of the test set with training data (“data leakage”) might go unnoticed. In this case, the model is effectively trained on part of the test set and estimates of the generalization error might be too optimistic and unreliable. Such a scenario can occur even when no obvious mistakes were made, for example, when the structures for a data set are sampled by running a long MD simulation where snapshots are written very frequently. Structures collected from adjacent time steps may be highly correlated in this case and when splitting the data randomly into training and test sets, a large portion of both sets will be almost identical. In such a case, instead of using a random split, a better approach would be to use a time-split of the data set,<sup>231</sup> for example, using the first 80% of the MD trajectory as the training/validation set and reserving the last 20% for testing.

**3.4.2. Data Preprocessing.** Prior to training a model, the raw data is often preprocessed in some way to improve the numerical stability of the ML algorithm. For example, a common practice is normalization, where inputs (or prediction targets) are scaled and shifted to lie in the range  $-1 \dots 1$  or to have a mean of zero and unit variance. The constants required for such transformations must never be extracted from the complete data set. Instead, only the training set may be used to obtain this information.<sup>30,90,231</sup> Otherwise, estimates of the generalization error on the test set may be overconfident (this is another form of data leakage). While normalization may be less common for the purpose of constructing ML-FFs, any “data-dependent” transformation must be done carefully. For example, it may be desirable to subtract the mean energy of structures from the energy labels to obtain numbers with smaller absolute values (for numerical reasons). This mean energy should be calculated only from the structures in the training set.

If a model is trained using a hybrid loss that incorporates multiple interdependent properties, such as energy and forces, it is important to consider the effects of the normalization procedure on the functional relationship of those values. For example, multiplying the energy labels by a factor requires that the forces are treated in the same way, because the factor carries over to the derivative (scaling energies and forces by different factors would therefore introduce inconsistencies in the data). Also, while subtracting the mean value from energy labels is valid, it is not correct to add any constant to the force labels, because that would translate into a linear term in the energy domain (the energy is related to the forces through integration). Consequently, the consistency between both label types would be broken and an energy conserving model would be incapable of learning. Even when doing simple unit transformations, care should be taken not to introduce any inconsistencies. For example, when energy labels are converted from  $E_h$  to  $\text{kcal mol}^{-1}$  and atom coordinates from  $a_0$  to  $\text{\AA}$ , force labels have to also be converted to  $\text{kcal mol}^{-1} \text{\AA}^{-1}$  so that all data is consistent. Depending on which code was used to obtain the reference data, it is even possible that units for some labels must be converted, because they may be given in different unit systems in the raw data (*ab initio* codes often report energy and forces in atomic units, whereas for coordinates, angstroms are popular).

### 3.5. Training the Model

After the data have been collected and prepared, the next step is training the ML-FF. During the training process, the

parameters of the model are tuned to minimize a loss function, which measures the discrepancy between the training data and the model predictions. In some cases, e.g., most kernel methods, the optimal solution can be found analytically. When this is not possible, for example, when training neural networks, the parameters are typically optimized iteratively by gradient descent or a similar algorithm. Because standard gradient descent tends to converge very slowly, some authors have proposed to augment it with terms mimicking momentum<sup>232,233</sup> or adaptive step sizes.<sup>234,235</sup> Not only training times, but also the achievable accuracy varies greatly with different optimization algorithms, so it is best to try different schemes (see ref 236 for an overview over different popular methods). A good default choice is the Adam optimizer,<sup>237</sup> which converges quickly and gives good results for many different NN architectures. The hyperparameters of a model (e.g., the number of layers or their width in the case of NNs) can also be selected in this step, albeit by checking the model performance on the validation set after training (instead of optimizing them directly). This section details the training process and highlights important points to consider, for example, the choice of loss function or how to prevent overfitting of the model to the training data.

**3.5.1. Choosing the Loss Function.** For regression tasks, a standard choice for the loss function is the mean squared error (MSE) given by  $\mathcal{L} = \frac{1}{N} \sum_{i=1}^N (y_i - \hat{y}_i)^2$ , because it punishes outliers disproportionately. Here, the index  $i$  runs over all  $M$  samples of the training data,  $y_i$  is the reference value for data point  $i$  and  $\hat{y}_i$  is the corresponding model prediction. When the MSE is used as loss function, it is implicitly assumed that any noise present in the reference data is distributed normally, which without additional information, is a sensible guess for most data. Further, the MSE loss allows finding the optimal parameters analytically (due to convexity) for linear ML algorithms, such as kernel ridge regression (see eq 10 in Section 2.2.1). However, the MSE is not necessarily the best choice for all cases. For example, to make the model less sensitive to outliers, a common alternative is to use a mean absolute error (MAE) loss given by  $\mathcal{L} = \frac{1}{N} \sum_{i=1}^N |y_i - \hat{y}_i|$ . Other functional forms, such as Huber loss<sup>238</sup> or even an adaptive loss,<sup>239</sup> are also possible, provided they are a meaningful measure of model performance.

After deciding on the general form of the loss function, the question remains which labels  $y$  to use as a reference. While the potential energy is an obvious choice, in classical MD, the PES is explored via integration of Newton’s second law of motion, which exclusively involves atomic forces. Since an important objective of ML-FFs is to reproduce the dynamical behavior of molecules in MD simulations as well as possible, it could even be argued that accurate force predictions should take priority over energy predictions in MD applications. However, since energy labels are usually available as a byproduct of force calculations, it seems reasonable to include both label types in the hope that this will help improve the overall prediction performance for both quantities. This gives rise to models based on hybrid loss functions that simultaneously penalize force  $F$  and energy  $E$  training errors. Assuming an MSE loss, it generally takes the form

$$\mathcal{L} = \frac{1}{N} \sum_{i=1}^N \frac{\|\hat{\mathbf{F}}_i - \mathbf{F}_i\|^2}{\mathcal{L}_{\mathbf{F}_i}} + \eta \frac{(\hat{E}_i - E_i)^2}{\mathcal{L}_{E_i}} \quad (27)$$



where the hyperparameter  $\eta$  determines the relative weighting between both loss terms to account for differences in units, information content, and noise level of the label types. A bilateral reduction of both loss terms is only possible if the objectives are noncompeting, that is, when the optimal parameter set is equally effective across both tasks. For this to be true, it must hold that

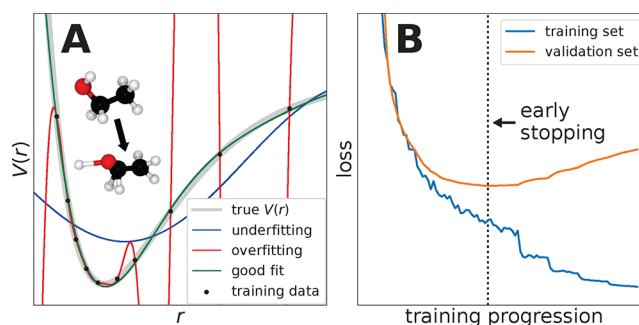
$$\begin{aligned} \|\hat{\mathbf{F}}_i - \mathbf{F}_i\|^2 &= \eta(\hat{E}_i - E_i)^2, \\ \Rightarrow \|\nabla_{\mathbf{r}}(\hat{E}_i - E_i)\|^2 &= \eta(\hat{E}_i - E_i)^2 \end{aligned} \quad (28)$$

at every training point  $i$  (here, the relation  $\mathbf{F} = -\nabla_{\mathbf{r}}E$  was substituted). Otherwise, the objectives  $\mathcal{L}_{\mathbf{F}}$  and  $\mathcal{L}_{E_i}$  are necessarily minimized by a different set of model parameters. Eq 28 is only true in general when  $\mathcal{L}_{E_i} = \mathcal{L}_{\mathbf{F}} = 0$  for all  $i$ , which is not fulfilled in practice because both labels may contain noise and they can usually not be fitted perfectly. A model trained using a hybrid loss (eq 27) will thus have to compromise between fulfilling both objectives on the training data, as opposed to joining energy and force labels for a performance gain on both. For this reason, the use of hybrid loss functions (or how to weight different contributions) warrants careful consideration depending on the intended application of the final model. Some models, for example, (s)GDML (see Section 2.3.2), do not even include energy constraints in their loss function at all and are trained on forces only. The energy can still be recovered via integration, but it does not participate in the training procedure except for determining the integration constant. In the end, the ultimate measure of a model's quality should not be how well it minimizes a particular loss function, but instead how well it is able to reproduce the experimental observables of interest. Also, it is important to keep in mind that the loss function measured on the training data is only a proxy for the true objective of any model, which is to generalize to unseen data. Compromising between the energy and force labels of the training data can even improve prediction accuracy for *both* label types on unseen data. For a more thorough discussion on the role of gradient reference data and how it can improve prediction performance, see ref 240–242.

**3.5.2. Tuning Hyperparameters.** Hyperparameters, such as kernel widths or the depth and width of a neural network, are typically optimized independently of the parameters that determine the model fit to the data: A hyperparameter configuration is chosen, the model is trained, and its performance is measured on the validation set. This process is repeated for as many trials as are affordable or until the desired accuracy is reached. Here it is crucial that no test data is used to measure model performance when tuning hyperparameters, so the ability to estimate the generalization error on the test set is not compromised. Choosing good values for the hyperparameter regimes requires some experience and intuition of the problem at hand. Fortunately, many models are quite robust and good default hyperparameters exist, which do not require any further tuning to arrive at good results. In other cases, hyperparameter tuning can be automated (for example via grid or random search<sup>30,90,150,165,243</sup>) and does not need to be performed manually. See also Section 2.2.3 for a more detailed discussion on tuning hyperparameters.

**3.5.3. Regularization.** Because ML models contain many parameters (sometimes even more than the number of data points used for training), it is possible or even likely that they

“overfit” to the training data. An overfitted model achieves low prediction errors on the training set, but performs significantly worse on unseen data (Figure 17A). The aim of regularization methods is to prevent this unwanted effect by limiting or decreasing the complexity of a model.



**Figure 17.** (A) One-dimensional cut through a PES predicted by different ML models. The overfitted model (red line) reproduces the training data (black dots) faithfully, but oscillates wildly in between reference points, leading to “holes” (spurious minima) on the PES. During an MD simulation, trajectories may become trapped in these regions and produce unphysical structures (inset). The properly regularized model (green line) may not reproduce all training points exactly, but fits the true PES (gray line) well, even in regions where no training data is present. However, too much regularization may lead to underfitting (blue line), that is, the model becomes unable to reproduce the training data at all. (B) Typical progress of the loss measured on the training set (blue) and on the validation set (orange) during the training of a neural network. While the training loss decreases throughout the training process, the validation loss saturates and eventually increases again, which indicates that the model starts to overfit. To prevent overfitting, the training can be stopped early once the minimum of the validation loss is reached (dotted vertical line).

When the loss function is minimized iteratively by gradient descent or similar algorithms, as is common practice for training NNs, one of the most simple methods to prevent overfitting is early stopping<sup>244</sup>). In the beginning of the training process, prediction errors typically decrease on both training and validation data. At some point, however, because the validation set is not used to directly optimize parameters, the performance on the training data will continue to improve, whereas the loss measured on the validation set will stagnate at a constant value or even begin to increase again. This indicates that the model starts overfitting. Early stopping simply halts the training process as soon as the validation error converges (instead of waiting for convergence of the training error), see Figure 17B. Early stopping also limits the size of the neural network weights and thus implicitly limits the complexity of the underlying function class. Similar to tuning hyperparameters, only the validation set, but never the test set, must be used for determining the stopping point.

Another method of regularization is the introduction of penalty terms to the loss function. Since overfitted models often are characterized by high variance in the prediction (see Figure 17A), the idea is to penalize large model parameters. For example,  $L^2$  regularization (adding the squared magnitude of parameters to the loss) shrinks the  $L^2$ -norm of the parameter vectors toward zero and prevents very large parameter values. On the other hand,  $L^1$  regularization (adding the absolute values of parameters to the loss) shrinks their  $L^1$ -norm, that is, it favors sparse parameter combinations.

Typically, the regularization term is weighted by an additional hyperparameter  $\lambda$  that determines its strength (like all hyperparameters,  $\lambda$  has to be tuned on the validation set). Note that solving eq 10 to determine the parameters of a kernel method will result in an  $L^2$ -regularized model trained on the MSE loss function.

### 3.6. Using ML-FFs in Production

The main motivation for training an ML-FF is to use it for some production task such as running an MD simulation. Before doing so, however, it is advisable to verify that it fulfills the accuracy requirements for its intended application. At this point in time, the test set becomes important: Since it was neither used directly nor indirectly during the training process, the data in the test set allows to estimate the performance of a model on truly unseen data, that is, how well it generalizes. For this, it is common practice to compute summary errors on the test set, for example, the mean absolute error (MAE) or root mean squared error (RMSE), as a measure of the overall accuracy of a model. In general, such a way of quantifying accuracy gives an overview of the ML model's performance on the given data set and provides a simple way to benchmark.

However, summary errors are biased toward the densely sampled regions of the PES, whereas much larger errors can be expected for less populated regions. Therefore, while summary errors measured on the test set are typically a good indicator for the quality of a model, they are not necessarily the best way to judge how well an ML-FF performs at its primal objective, namely capturing the relevant quantum interactions present in the original molecular system. In other words, performance measures evaluated on the test set should not be trusted blindly. They are only reliable when the test set is representative of the new data encountered during production tasks, that is, when they are drawn from the same distribution. When a model has to extrapolate, it might give unreliable predictions, even when its performance on the test set is satisfactory. When in doubt, especially when an ML-FF is used for a different task than it was originally constructed for, it is better to collect a few new reference data points to verify that a model is still valid for its use case. Because of the generally limited extrapolation capabilities of ML models, results obtained from studies with ML-FFs should always be scrutinized more carefully than results obtained with conventional FFs. For example, it is advisable to randomly select a few trajectories and verify that the sampled structures look "physically sensible", for example, no extremely short or long bonds are present and atoms have no unusual valencies. Since the PES is a high-dimensional object, rare events, where a trajectory visits a part of configurational space that is not sampled in the reference data, are always possible, even when the PES was carefully sampled. If any questionable model predictions are found, it is advisable to double-check their accuracy with additional reference calculations.

### 3.7. Example Code and Software Packages

While many modern ML-FFs are conceptually simple, their implementation is often not straightforward, involving many intricate details that can not be exhaustively covered in publications. Instead, those details are best conveyed by a reference implementation of the respective model. Publicly available well maintained codes allow to replicate numerical experiments and to build on top of existing models with minimal effort.

In this section, example code snippets for training and evaluating kernel- and NN-based ML-FFs with the sGDML<sup>165</sup> (Section 3.7.1) and SchNetPack<sup>245</sup> (Section 3.7.2) software packages are given. This is followed by a short description of other popular software packages for the construction of ML-FFs (Section 3.7.3) as a first orientation for interested readers. Note that the list is not comprehensive and many other similar packages exist.

**3.7.1. The sGDML Package.** A reference implementation of the (s)GDML model is available as Python software package at <http://www.sgdml.org>.<sup>165</sup> It includes a command-line interface that guides the user through the complete process of model creation and testing, in an effort to make this ML approach accessible to broad practitioners. Interfaces to the Atomic Simulation Environment (ASE)<sup>246</sup> or i-PI<sup>247</sup> make it straightforward to perform MD simulations, vibrational analyses, structure optimizations, nudged elastic band computations, and more.

To get started, only user-provided reference data is needed, specifically a set of Cartesian geometries with corresponding total energy and atomic-force labels. Force labels are necessary, because sGDML implements energy conservation as an explicit linear operator constraint by modeling the FF reconstruction as the transformation of an underlying energy model (see Section 2.3.2). The trained model will give predictions at the accuracy of the reference data and can be queried like any other FF.

**3.7.1.1. Data Set Preparation.** The sGDML package uses a proprietary format for its data sets, but scripts to import and export to all file types supported by the ASE package,<sup>246</sup> which covers most popular standards, are included. To convert a <dataset>, simply call `sgdml_dataset_via_ase.py <dataset>` and follow the instructions.

**3.7.1.2. Training.** The most convenient way to reconstruct a FF is via the command line interface: `sgdml all <dataset> <ntrain> <nvalid>`. This command will automatically generate a fully trained and cross-validated model and save it to a file, that is, model selection and hyperparameter tuning (see Section 2.2.3) are performed automatically. The parameters <ntrain> and <nvalid> specify the sample sizes for the training and validation subsets, respectively. All remaining points are reserved for testing. Each subset is sampled from the provided reference <dataset> without overlap.

**3.7.1.3. Using the Model.** To use the trained model, the sGDML predictor is instantiated from the <model> file generated above and energy and forces are queried for a given geometry (for example stored in an XYZ file <xyz>):

```

import numpy
from sgdmllib.predict import GDMLPredict
from sgdmllib.utils import io

# Load model from file
parameters = numpy.load("<model>")
gdml = GDMLPredict(parameters)

# Load structure from xyz file
r,_ = io.read_xyz("<xyz>")

# Evaluate model
# (energies e and forces f)
e,f = gdml.predict(r)

```

It is also possible to run MD simulations using ASE and the Calculator interface included with the sGDML package:

```

import ase
from sgdmllib.intf.ase_calc import \
    SGDMLCalculator
from ase.md.velocitydistribution import \
    MaxwellBoltzmannDistribution

# Load sGDML model as ASE calculator
calc = SGDMLCalculator("<model>")

# Load structure and attach calculator
atoms = ase.io.read("<xyz>")
atoms.set_calculator(calc)

# Initialize momenta at 300 K
MaxwellBoltzmannDistribution(
    atoms, 300*units.kB)

# Setup MD using the velocity Verlet
# integrator and a time step of 0.2 fs
dyn = ase.md.verlet.VelocityVerlet(
    atoms, 0.2*ase.units.fs,
    trajectory="<trajectory>")

# Simulate for 1000 steps
dyn.run(1000)

```

To run this script, a trained model (<model>) and an initial geometry (<xyz>) are needed. The resulting MD trajectory is stored in a file <trajectory>. For more details and applications examples, please visit the documentation at [www.sgdmllib.org/doc/](http://www.sgdmllib.org/doc/).

**3.7.2. The SchNetPack Package.** SchNetPack<sup>245</sup> is a toolbox for developing and applying deep neural networks to the atomistic modeling of molecules and materials available from <https://schnetpack.readthedocs.io/>. It offers access to models based on (weighted) atom-centered symmetry functions and the deep tensor neural network SchNet, which can be coupled to a wide range of output modules to predict potential energy surfaces and forces, as well as a growing number of other quantum-chemical properties. SchNetPack is designed to be readily extensible to other neural network potentials such as the DTNN<sup>160</sup> or PhysNet.<sup>108</sup> It provides extensive functionality for training and deploying these models, including access to common benchmark data sets. It also provides an Atomic Simulation Environment (ASE)<sup>246</sup>

calculator interface, which can be used for performing a wide variety of tasks implemented in ASE. Moreover, SchNetPack includes a fully functional MD suite, which can be used to perform efficient MD and PIMD simulations in different ensembles.

As it is based on the PyTorch deep learning framework,<sup>248</sup> SchNetPack models are highly efficient and can be applied to large data sets and across multiple GPUs. Combined with the modular design paradigm of the code package, these features also allow for a straightforward implementation and evaluation of new models. Similar to the sGDML package, the central commodity for training models in SchNetPack is a data set containing the Cartesian geometries (including unit cells and periodic boundary conditions, if applicable) and atom types, as well as the target properties to be modeled (e.g., energies, forces, dipole moments, etc.). More information can be found in ref 245.

**3.7.2.1. Data Set Preparation.** SchNetPack uses an adapted version of the ASE database format to handle reference data. The package provides several routines for preparing custom data sets, as well as a range of preconstructed data set classes for popular benchmarks (e.g., QM9<sup>249</sup> and MD17<sup>105</sup>), which will automatically download and format the data. For example, molecular data from the MD17 data set can be loaded via `spk_load.py md17 <molecule> <path>` where <molecule> indicates the molecule for which data should be loaded (e.g., ethanol), while the second argument specifies where the data is stored locally.

SchNetPack also provides a utility script for converting data files in the extended XYZ format, which is able to handle a wide variety of properties, to the database format used internally. Conversion can be invoked with the command `spk_parse.py <input> <target>` where the arguments specify the file paths to the <input> data file and <target> database in SchNetPack format, respectively.

**3.7.2.2. Training.** As for the sGDML package, training and evaluating ML models in SchNetPack can be performed via a command line interface. For example, a basic model can be trained with the script: `spk_run.py train [model_type] [dataset_type] <dataset> <model> --split <ntrain> <nvalid>`. Here, [model\_type] specifies which kind of NNP to use (wacsf for a descriptor-based NNP using wACSFs,<sup>206</sup> or schnet for the SchNet<sup>109</sup> end-to-end NNP architecture) and [dataset\_type] specifies either a preexisting data set (e.g., qm9 or md17), or a custom data set provided by the user. The next two arguments are the paths to the reference <dataset> and the file the trained <model> will be written to. The arguments <ntrain> and <nvalid> specify the sample sizes for the training and validation subsets, while the remaining points are reserved for testing. SchNetPack offers a wide range of additional settings to modify the training process (e.g., model composition, use of GPU, how different properties should be treated etc.), see <https://schnetpack.readthedocs.io/>.



**3.7.2.3. Using the Model.** Once a model has been trained, it can be evaluated in several different ways. The most basic method is to perform predictions via:

```
import torch
import ase
from schnetpack.data.atoms \
import AtomsConverter

# Load model from file
spk_model = torch.load("<model>")

# Set up converter for ASE atoms
converter = AtomsConverter()

# Load structure from xyz file
atoms = ase.io.read("<xyz>")
inputs = converter(atoms)

# Evaluate model and collect
# predictions in a dictionary
results = spk_model(inputs)
```

It is also possible to use the SchNetPack MD suite to perform various simulations with the trained model. Continuing the above example, a basic MD run can be carried out as

```
import ase
import schnetpack.md as md

# Set up the system using 1 replica
# (use n_replicas > 1 for PIMD)
md_system = md.System(n_replicas=1)
# Load structure from xyz file
atoms = ase.io.read("<xyz>")
md_system.load_molecules(atoms)

# Initialize momenta at 300 K
init = md.MaxwellBoltzmannInit(300)
init.initialize_system(md_system)

# Set up integrator
tstep = 0.2 # fs
integrator = md.VelocityVerlet(tstep)

# Prepare model for MD (specify
# units, required properties, etc.)
calculator = md.SchnetPackCalculator(
    spk_model,
    required_properties=["forces"],
    force_handle="forces",
    position_conversion='A',
    force_conversion='kcal/mol/A')

# Combine everything
simulator = md.Simulator(
    md_system, integrator, calculator)

# Simulate for 1000 steps
simulator.simulate(1000)
```

Simulations can be further modified via hooks, which introduce temperature and pressure control, as well as various sampling schemes. Further documentation of the code package and usage tutorials can be found at <https://schnetpack.readthedocs.io/>.

**3.7.3. Other Software Packages.** **3.7.3.1. AMP: Atomistic Machine-Learning Package.** AMP is a Python package designed to integrate closely with the Atomistic Simulation Environment<sup>246</sup> (ASE) and aims to be as intuitive as possible. Its modular architecture allows many different combinations of structural descriptors and model types. The main idea of AMP is to construct ML-FFs *on-demand*, that is, simulations are first started with an *ab initio* method and later switched to the ML-FF once the model is sufficiently accurate. The package is described in greater detail in ref 191 and on its official website <https://amp.readthedocs.io/>.

**3.7.3.2.  $\text{aenet}$ .** The Atomic Energy NETwork ( $\text{aenet}$ ) package includes tools for constructing and applying neural network-based ML-FFs. It is written in Fortran 95/2003 and utilizes efficient BLAS (Basic Linear Algebra Subprograms) and LAPACK (Linear Algebra PACKage) routines for performing linear algebra. A Python interface is also included. More details can be found on <https://github.com/atomisticnet/aenet/>.

**3.7.3.3. DeePMD-Kit.** The DeePMD-kit is a package written in Python/C++ aiming to minimize the effort required to build deep NNPs with different structural descriptors. It is based on the TensorFlow deep learning framework<sup>250</sup> and offers interfaces to the high-performance classical and path-integral MD packages LAMMPS<sup>251</sup> and i-PI.<sup>247</sup> More details on the DeePMD-kit can be found in ref 252 or on <https://github.com/deepmodeling/deepmd-kit/>.

**3.7.3.4. Dscribe.** Dscribe is a Python package for transforming atomic structures into fixed-size numerical fingerprints.<sup>253</sup> These descriptors can then be used as input for neural networks or kernel machines to construct ML-FFs. Supported representations include the standard Coulomb matrix<sup>28</sup> and variants for the description of periodic systems,<sup>119</sup> ACSFs,<sup>116</sup> SOAP,<sup>117</sup> and MBTR.<sup>170</sup> More details can be found on the official Web site <https://singroup.github.io/dscribe/> or in ref 253.

**3.7.3.5. n2p2.** The neural network potential package (n2p2) allows researchers to use existing parametrizations of Behler-Parinello NNPs to predict energies and forces (either with standalone tools or with the LAMMPS MD package<sup>251</sup>), but it also provides training tools for generating new potentials. It is mainly written in C++. For further details, refer to <https://compphysvienna.github.io/n2p2/>.

**3.7.3.6. PROPhet.** The PROPhet Prophet (or short: PROPhet) package uses neural networks to predict the relationship between chemical structure and material properties. As such, it can also be used to generate NN-based ML-FFs. It includes tools to automatically extract properties of interest from the output files of several *ab initio* codes and an interface to the LAMMPS MD package.<sup>251</sup> More details can be found on <https://bikloost.github.io/PROPhet/>.

**3.7.3.7. QML.** QML is a toolkit for learning properties of molecules and solids written in Python.<sup>254</sup> It supplies building blocks to construct efficient and accurate kernel-based ML models, such as different kernel functions and premade implementations of many different structural representations, for example, Coulomb matrix,<sup>28</sup> SLATM,<sup>131</sup> and FCHL.<sup>107</sup> The package is primarily intended for the general prediction of

Table 1. Overview of Different Topics and Applications of ML-FFs Discussed in This Section<sup>a</sup>

category	ref	ML-FF	max. $N_{\text{atoms}}$	reference theory
electronic effects	Sauceda et al. <sup>70</sup>	sGDML	21	CCSD(T), CCSD
	Sauceda et al. <sup>214</sup>	sGDML	21	CCSD(T), CCSD
	Sauceda et al. <sup>69</sup>	sGDML	21	PBE, CCSD(T), CCSD
thermodynamics	Morawietz et al. <sup>260</sup>	BP-NNP	6912	RPBE, BLYP
	Andrade et al. <sup>261</sup>	DeepMD	426	SCAN
	Deringer et al. <sup>262</sup>	GAP	1000	LDA
	Behler et al. <sup>263</sup>	BP-NNP	64	LDA
	Bartok et al. <sup>264</sup>	GAP	23 496	PW91
	Deringer et al. <sup>265</sup>	GAP	4096	PW91
	Bonati et al. <sup>266</sup>	DeepMD	680	SCAN
	Brickel et al. <sup>213</sup>	PhysNet	6	MP2
	Unke et al. <sup>267</sup>	RKHS	3	UCCSD(T)
reactions	Denis et al. <sup>268</sup>	RKHS	3	UCCSD(T)-F12a
	Lu et al. <sup>269</sup>	PIP-NN	7	UCCSD(T)-F12a
	Sweeny et al. <sup>270</sup>	PhysNet	7	MP2
	Käser et al. <sup>271</sup>	PhysNet	7	MP2
	Rivero et al. <sup>272</sup>	PhysNet	19	M06-2X
	Liu et al. <sup>273</sup>	BP-NNP	38	RPBE
	Chmiela et al. <sup>105</sup>	GDML	21	PBE
	Chmiela et al. <sup>69</sup>	sGDML	21	CCSD, CCSD(T)
	Schütt et al. <sup>274</sup>	SchNet	20	PBE
nuclear quantum effects	Sauceda et al. <sup>161</sup>	sGDML	21	CCSD, CCSD(T)
	Hellström et al. <sup>275</sup>	BP-NNP	1700	RPBE
	Chen et al. <sup>276</sup>	HDNN	5	CASSCF
	Westermayr et al. <sup>277</sup>	NN	6	MR-CISD
	Westermayr et al. <sup>278</sup>	SchNet	6	MR-CISD, CASSCF
excited states	Gastegger et al. <sup>71</sup>	BP-NNP	209	BLYP, BP86, B2PLYP
	Yao et al. <sup>195</sup>	BP-NNP	60	$\omega$ B97X-D
spectroscopy	Raimbault et al. <sup>279</sup>	SOAP	80	PBE
	Sommers et al. <sup>280</sup>	DeepMD	512	SCAN

<sup>a</sup>In all cases, the type of employed ML-FF is given along with the number of atoms of the largest system used to study the respective phenomenon. The basic level of reference theory (neglecting basis sets and dispersion corrections for clarity) is also reported.

chemical properties but can also be used for the construction of ML-FFs. For further details, refer to the official Web site <https://www.qmlcode.org> or the github repository <https://github.com/qmlcode/qml/>.

**3.7.3.8. RKHS Toolkit.** The RKHS toolkit is mainly intended for constructing highly accurate and efficient PESs for studying scattering reactions of small molecules. As described in section 2.2.1, the evaluation of kernel-based methods scales linearly with the number of training points  $M$  (see eq 2). By using special kernel functions and precomputed lookup tables, the RKHS toolkit allows researchers to bring this cost down to  $O(\log N)$ . However, it requires that the training data has grid structure, which limits its applicability to small systems, where it is meaningful to sample the PES by scanning a list of values for each internal coordinate. The implemented kernel functions also allow to encode physical knowledge about the long-range decay behavior of certain coordinates, which enables accurate extrapolation well beyond the range covered in the training data. A Fortran90 implementation of the toolkit can be downloaded from <https://github.com/MMunibas/RKHS/> and the algorithmic details are described in ref 255.

**3.7.3.9. RuNNer.** The RuNNer Code was the first implementation of high-dimensional neural network potentials and the source code is freely available. Details on how to obtain access can be found on <https://www.uni-goettingen.de/de/560580.html>.

**3.7.3.10. TensorMol.** The TensorMol package allows researchers to train NNPs that explicitly account for electrostatic interactions. It is based on the TensorFlow deep learning framework<sup>250</sup> and includes an interface to i-PI<sup>247</sup> for performing path integral simulations. For further information, refer to ref 195 or <https://github.com/jparkhill/TensorMol>.

## 4. PHYSICAL AND CHEMICAL INSIGHTS FROM MACHINE LEARNED FORCE FIELDS

In nature, the atoms in chemical systems are in constant motion, giving rise to various configurations and reactive events. A large number of experimental observations are not based on a single molecule or atom, but instead on ensembles of various species subject to external conditions, such as temperature or pressure. Consequently, properties associated with individual structures are not sufficient to characterize macroscopic systems. One way to compute ensemble averages are molecular dynamics (MD) simulations, where the time evolution of a system is governed by the atomic forces derived from its associated potential energy surface (PES). From the ergodic hypothesis<sup>256</sup> it is known that the expected value of an observable  $A$  can also be obtained from the time average  $\langle A \rangle_t = T^{-1} \sum_{i=1}^T A_i$ , where  $A_i$  is the value of  $A$  corresponding to the structure at time step  $t$  of the dynamics trajectory and  $T$  is their total number. Of course, this relation is valid only when

the dynamics is long enough to visit all configurations of the system accessible under the simulation conditions.

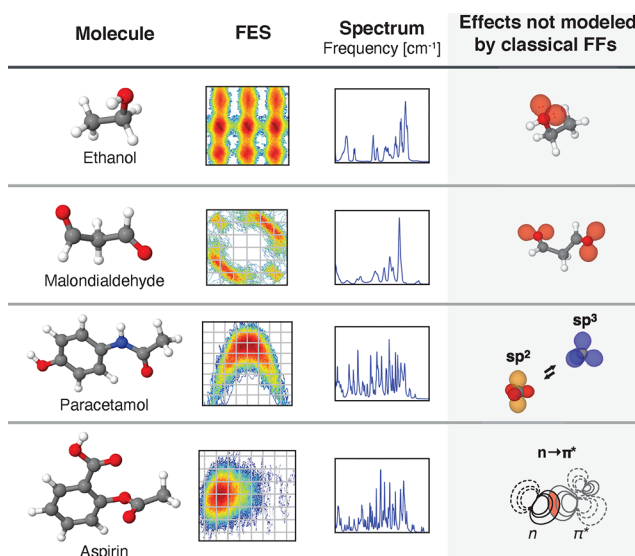
To obtain meaningful statistics with MD simulations, many thousands (or millions) of successive PES evaluations are necessary. Because of their high computational cost, accurate electronic structure PESs quickly become intractable for such simulations, which is why highly efficient classical force fields (FFs) are usually employed for running MD simulations. However, this efficiency comes at a cost: Conventional FFs completely neglect or misrepresent some potentially relevant contributions to the potential energy, such as polarization, charge transfer, or electronic effects, which limits their usefulness in modeling complex chemical phenomena. Machine-learned FFs (ML-FFs) offer a unique combination of computational efficiency and high accuracy, opening up tantalizing new possibilities in the simulation of the dynamics of molecules, surfaces, materials and condensed phases. They are able to model *all* chemical interactions—including those that are typically neglected by conventional FFs. The high accuracy of ML-FFs allows to obtain qualitatively different and novel insights, which would otherwise only be accessible from computationally infeasible *ab initio* MD (AIMD) simulations. In the following, some chemical insights made possible by ML-FFs, which could not have been obtained with conventional FFs, are highlighted in greater detail. A brief overview is given in Table 1. Note that the given examples represent only a tiny fraction of the published literature, an exhaustive list is beyond the scope of the current review. Interested readers can find further examples in other review articles, for example, in refs 257–259.

#### 4.1. Electronic Effects

A good example for the power of ML-FFs is a recent study of the dynamics of small molecules (malondialdehyde, ethanol, salicylic acid, paracetamol, aspirin) with atomic forces at CCSD(T) quality.<sup>70</sup> AIMD simulations were run at 500 K at the PBE+TS/DFT level of theory<sup>167,168</sup> and the collected configurations randomly subsampled to calculate energies and forces at the CCSD(T) level of theory (reference data for aspirin was calculated at CCSD accuracy). For each molecule, an ML-FF was constructed from 1000 data points with the sGDML<sup>165</sup> method (see Section 2.3.2) and used to run MD simulations at 300 K. Running simulations of this quality with *ab initio* methods is impossible, as they would require up to a billion times more computation time. Conventional FFs were shown to be no viable alternative to ML-FFs, as they do not adequately describe, or even completely neglect, effects which strongly influence the dynamics, and hence the properties, of the studied molecules (Figure 18).

For example, in ethanol, the lone pairs of the oxygen atom interact with the partially positively charged hydrogen atoms of the methyl group. Because of this attraction, the configuration where both lone pairs are adjacent to a hydrogen atom is visited most frequently during a dynamics simulation. Any derived property, for example, the Gibbs/Helmholtz free energy surface (FES) or the infrared spectrum, is only accurate when this effect is properly described. Conventional FFs do not account for lone pairs and are thus unable to predict the molecular properties correctly.

A similar effect can be observed in malondialdehyde. Here, the lone pairs of the two oxygen atoms strongly repel each other, which drives the dynamics away from configurations where they are close. While conventional FFs can crudely



**Figure 18.** Visualization of electronic effects, which are accurately modeled by ML-FFs, but neglected by conventional FFs. Electron lone pairs, hybridization changes and orbital donation effects all influence the dynamics of molecules and hence the properties that are computed from MD simulations. When predicting, for example, Gibbs/Helmholtz free energy surfaces (FESs) or molecular spectra, neglecting them will lead to qualitatively different results.

model electrostatic repulsion between the oxygen atoms with negative partial charges, the steric contributions from the overlap of the electron clouds is not described, causing a qualitatively different dynamics.

Paracetamol is another molecule where lone pairs influence the stability of specific configurations: The partially positively charged phenyl-hydrogen adjacent to the oxygen atom of the acetamide group interacts with its lone pairs and favors a specific dihedral angle. Additionally, the nitrogen atom of the acetamide group is  $sp^2$  hybridized, which allows conjugation to the electrons in the phenyl system and leads to the planar geometry of paracetamol. When the nitrogen hybridization state is changed to  $sp^3$ , the energetically favorable interaction is broken and corresponding configurations are thus rarely visited during room temperature dynamics. However, at higher temperatures, the hybridization state may switch frequently—conventional FFs are unable to describe this.

Another important electronic effect can be observed in aspirin. Here, an occupied (lone pair)  $n$  orbital of the carbonyl group overlaps with an unoccupied antibonding  $\pi^*$  orbital in the ester group. This  $n \rightarrow \pi^*$  interaction dictates the relative arrangement of these functional groups in the global minimum structure.<sup>214</sup> The effect is even amplified during dynamics since thermal fluctuations enhance the overlap.<sup>69</sup>

These and many other electronic effects, for example,  $n \rightarrow \sigma^*$  interactions, hyperconjugation, and Jahn–Teller distortions, are captured automatically by ML-FFs. In contrast, including them in conventional FFs would require additional terms, whose functional form (and even which effects need to be modeled) are typically unknown *a priori*.

#### 4.2. Thermodynamics

A typical application of classical FFs is the study of thermodynamic properties of bulk systems, such as enthalpies, entropies, and phase diagrams. However, their limited accuracy is a major obstacle for quantitative predictions, as small



inaccuracies in the interaction of a few particles will inevitably lead to big discrepancies when studying many particles. A good example for this are van der Waals (vdW) interactions. They are weak contributions to the total potential energy for small molecules in gas phase, but they add up in large condensed systems and bulk materials and can strongly influence their properties and dynamics.<sup>281</sup> While conventional FFs account for vdW interactions, they typically do so with a relatively crude model based on the Lennard-Jones potential,<sup>14</sup> which is insufficient for quantitative predictions in many cases. A prime example is water: It is the most studied liquid in literature and many different conventional FFs for water (some with additional special-purpose terms) have been proposed in the last decades, yet none of them is able to reproduce all experimentally measured properties of water in MD simulations.<sup>282</sup>

Here, ML-FFs offer a promising alternative. Morawietz et al.<sup>260</sup> trained a descriptor-based NNP on periodic configurations of liquid and crystalline water, for which reference data was calculated with different DFT functionals. MD simulations with the ML-FF revealed that the thermodynamic anomalies of water, such as its density maximum and negative volume of melting, are due to a delicate balance of weak vdW forces. The study was able to accurately predict experimentally measured radial distribution functions, as well as temperature dependent shear viscosities and diffusion coefficients. As ML-FFs are naturally able to describe bond breaking and formation, the study could even investigate proton transfer between different water molecules.

The ability to analyze thermodynamic properties of reactive events is a major advantage of ML-FFs over conventional methods. For example, a recent study investigated the Gibbs free energy of proton transfer in liquid water at a titanium oxide surface.<sup>261</sup> A descriptor-based NNP was trained using reference data collected through an adaptive sampling approach and used to run MD simulations. The study revealed that a significant fraction of water molecules forms short-lived hydroxyl groups on the titanium oxide surface, which strongly influence its surface chemistry. Such insights are key to understanding phenomena such as surface functionalization and photocatalytic processes.

Another application where the flexibility of ML-FFs is a major advantage is the modeling of bulk materials. For example, Gaussian approximation potentials (GAPs, see Section 2.3.3) and NNPs have been constructed for elemental carbon<sup>262</sup> and silicon.<sup>263–266</sup> They allow researchers to investigate a wide range of phenomena of liquid, crystalline, and amorphous solid phases including defects and crack propagation. Modeling these effects accurately is only possible with ML-FFs or prohibitively expensive AIMD simulations. It is even possible to predict accurate phase diagrams of such systems with ML-FFs.<sup>263,264</sup> Since this requires a correct model of bond formation and breaking, as well as changes of bonding patterns, such insights could not be obtained from conventional FFs.

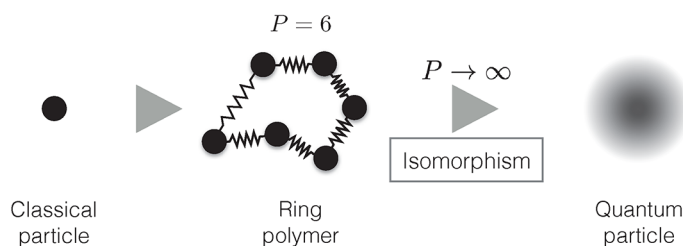
### 4.3. Reactions

One of the most significant advantages of ML-FFs over conventional FFs is their natural ability to model chemical reactions. Even in cases where it is possible to construct special purpose classical FFs that are able to describe reactions, they are typically much less accurate than their ML-FF counterparts. For example, a recent study compared an ML-FF

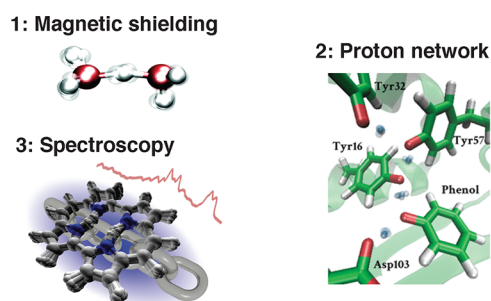
constructed with a message-passing NNP with two classical methods to obtain a reactive FF for the  $\text{Cl}-\text{CH}_3-\text{Br}$  transformation.<sup>213</sup> Here, the ML-FF achieved up to 3 orders of magnitude lower errors and yielded qualitatively and quantitatively different predictions for the Helmholtz free energy surface along the reaction path. It is therefore no surprise that one of the first fields where ML-FFs were employed with great success are reaction dynamics. Here, the chemical transformations associated with molecular collisions over short time and length scales are studied. These simulations offer detailed atomistic insights into the reaction mechanism, providing access to rate constants and scattering cross-sections, as well as insights on how the molecular energy is distributed between different modes, all of which can be directly related to experiments. To yield quantitative predictions, sufficient statistics and highly accurate PESs are required, making them an excellent application for ML-FFs. Studies typically involve small molecular systems, which are treated at high levels of accuracy, such as the collision of  $\text{N}_2^+$  and Ar<sup>267,268</sup> or the  $\text{Cl}+\text{CH}_3\text{OH} \rightarrow \text{HCl}+\text{CH}_3\text{O}/\text{CH}_2\text{OH}$  reaction.<sup>269</sup> Typical conventional FFs require fixed bonding patterns and are thus intrinsically unsuited for studying chemical reactions. While there also exist reactive variants of classical FFs, they do not reach the accuracy of ML-FFs. For example, a recent study investigated the thermal activation of methane by  $\text{MgO}^+$  with a message-passing NNP (see Section 2.3.4) and a reactive classical FF.<sup>270</sup> Here, the ML-FF achieved prediction errors up to two orders of magnitude lower than the classical variant compared to *ab initio* data. In addition, the disagreement between experimental rate constants and those predicted from MD simulations was lower by a factor of two with the ML-FF compared to the values obtained from the classical FF. The remaining discrepancy between prediction and experiment was further investigated and it was determined that the deviation was not due to inaccuracies of the ML-FF *per se* but instead could be traced back to the multireference character of the transition state, that is, problems with the *ab initio* reference data itself.

Even though it is possible to construct classical reactive FFs for specific reactions, there are cases where this is exceedingly difficult. A good example is a recent study where the phototautomerization reaction of acetaldehyde was investigated, which is speculated to be a major pathway for formic acid formation in the atmosphere.<sup>271</sup> After being photoexcited, acetaldehyde contains enough energy that it may not only tautomerize to ethenol, but also dissociate into carbon monoxide and methane, or into hydrogen and ethenone. An accurate description of all three possible reaction pathways with the same FF is extremely difficult to achieve with conventional methods. The NNP used for the study on the other hand was trained on MP2/aug-cc-pVTZ<sup>283,284</sup> reference data and allowed an unbiased description of all relevant processes at *ab initio* quality. Analyzing a total of 12 000 individual trajectories, the study concluded that the formation of ethenol from phototautomerization of acetaldehyde is unlikely under atmospheric conditions. This insight could not have been obtained by running AIMD simulations in a reasonable time frame: The combined simulation time of 1  $\mu\text{s}$  would amount to ten billion single point calculations (a time step of  $\Delta t = 0.1$  fs was used due to the large excitation energies). In contrast, less than 500k structures were used for training the ML-FF, that is, the time spent for running *ab initio*

## A) Path Integral Molecular Dynamics

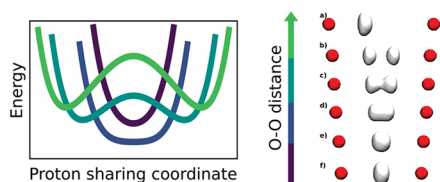


## B) PIMD results using DFT/CC

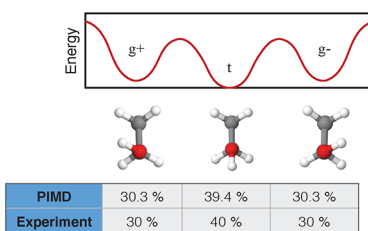


## C) PIMD results using ML-FFs

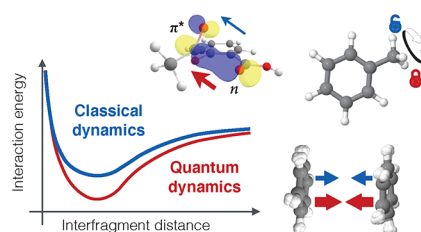
## 1: Ultra-low temperatures



## 2: Population analysis



## 3: Interactions enhancement



**Figure 19.** (A) Schematic description of the path integral (ring polymer) molecular dynamics (PIMD) method, where quantum particles are approximated by a classical ring polymer with  $P$  beads. There is an exact isomorphism between these two systems when  $P \rightarrow \infty$ , that is, their statistical properties become equivalent. (B) PIMD simulations using DFT or coupled cluster calculations. (1) Coupled cluster PIMD simulations of the Zundel model to compute  $^1\text{H}$  magnetic shielding tensor (adapted with permission from ref 287. Copyright 2015 published by the PCCP Owner Societies under CC BY-NC 3.0 <https://creativecommons.org/licenses/by-nc/3.0/>). (2) Example of hydrogen-bond networks and their NQE implications on biological functions and enzyme catalysis (adapted with permission from ref 288. Copyright 2017 American Chemical Society.). (3) IR spectrum of the porphycen molecule computed from PIMD simulations (adapted with permission from ref 289. Copyright 2019 American Chemical Society.). (C) PIMD simulations using ML-FFs trained on DFT or coupled cluster data. (1) Ultralow temperature dynamics of the Zundel model obtained from PIMD simulations (adapted with permission from ref 290. Copyright 2018 American Chemical Society.). (2) Comparison of the statistical sampling of different conformers of ethanol between experiment and simulations (adapted with permission from ref 69. Copyright 2018 Chmiela et al.). (3) Schematic description of the enhancement in intra- and intermolecular interactions due to NQEs (adapted with permission from ref 161. Copyright 2020 Saucedo et al.).

calculations was reduced by more than five orders of magnitude by employing an ML-FF.

Because of the efficiency of ML-FFs, scattering simulations can now even be extended to involve larger organic molecules. For example, a study of the minimum dynamic path<sup>285</sup> of Diels–Alder reactions of 1,3-dibromo-1,3-butadiene and maleic anhydride with an end-to-end NNP has revealed that molecular rotations are a major driving force for the formation of products,<sup>272</sup> an effect which had not been described previously in the literature for this type of reaction. ML-FFs can even be applied to reactions between molecules and surfaces. For example, a study by Liu et al.<sup>273</sup> investigated (reactive) HCl scattering on a gold surface using a descriptor-based NNP.

For a recent review on neural network-based PESs for small molecules and reactions, see ref 286.

#### 4.4. Nuclear Quantum Effects

Predictive simulations of molecular systems and materials require not only highly accurate representations of the potential energy surface (PES) but also appropriate statistical sampling of the PES. While classical MD simulations are sufficient for this in some cases, the quantum nature of nuclei plays an important role in many systems. Nuclear quantum fluctuations are a fundamental phenomenon in nature resulting from Heisenberg's uncertainty principle;<sup>291</sup> hence, physical and chemical properties of molecular or biological systems, as well

as nano- and bulk-materials, may be affected by them up to certain extent. In particular, light elements, such as protons and atoms in the first row of the periodic table, are prone to display nuclear quantum effects (NQEs) even at room temperature. Furthermore, materials or molecules formed by heavier atoms, but having strong bonds or being at low temperatures, exhibit significant NQEs.<sup>292–299</sup>

Consequently, to generate predictive simulations of many physical properties, NQEs must be incorporated. A widely used methodology to perform quantum dynamics is path integral molecular dynamics (PIMD). This method is based on the isomorphism between a quantum particle and a classical harmonic ring polymer of  $P$  beads (i.e.,  $P$  harmonically coupled copies of the particle), where the equality holds for  $P \rightarrow \infty$  (see Figure 19A).<sup>82</sup> In practice, convergence of thermodynamical properties can be achieved using only a small number of beads. For light atoms at room temperature for example,  $P \approx 16$ –32 is often sufficient to converge mechanical properties.<sup>165,287,289,297</sup> This number can be reduced even further by using more sophisticated thermodynamic estimators.<sup>300</sup>

Given that PIMD simulations require energies and forces for  $P$  copies of the system of interest, it is infeasible to use *ab initio* methods to derive them in most cases. There are some exceptions. For example, PIMD simulations to study the IR spectrum of the porphycen molecule have been performed using DFT with the B3LYP functional, and it was shown that

the correct Helmholtz free energy and vibrational spectrum can only be recovered by considering NQEs (see Figure 19B:3).<sup>289</sup> Another example includes PIMD simulations of the Zundel model at the CCSD level of theory to study the impact of NQEs on its structure and the <sup>1</sup>H magnetic shielding tensor.<sup>287</sup> However, both of these studies required supercomputers to make the calculations possible in a reasonable time frame. On the other hand, ML-FFs can replicate the same results at a fraction of the computational cost, that is, speed-ups by a factor of 10<sup>5</sup>–10<sup>7</sup> (depending on the reference level of theory) can be achieved.<sup>69,109</sup> This gain in computational efficiency makes it possible to run PIMD simulations for a wide range of systems and offers the chance to reveal new chemical and physical insights.

For example, Chmiela et al.<sup>105</sup> performed room temperature PIMD simulations of aspirin using a GDML model (see Section 3.7.1) trained on PBE+TS<sup>167,168</sup> reference data to investigate the paths followed between different minima on its PES. In a followup study, Chmiela et al.<sup>69</sup> compared free energies and vibrational density of states of a variety of medium-sized molecules obtained from PIMD simulations with a model trained on CCSD or CCSD(T) reference data to the same quantities obtained from a model trained on PBE+TS/DFT data. The authors found that even though the PESs at the two different levels of theory are very similar, tiny differences may still lead to largely different free energies. Additionally, it was shown that the experimentally determined populations for different conformations of ethanol can only be recovered from simulations when including NQEs (see Figure 19C:2).

In another study, Schütt et al.<sup>274</sup> investigated the dynamics of C<sub>20</sub> fullerene using a NNP trained on PBE+TS/DFT reference data. Here, including NQEs broadens the radial distribution function significantly, which also increases the molecular polarizability.<sup>161</sup> A change in the distribution of interatomic distances also influences electronic effects: A recent study of ref 70. (mentioned earlier in the paragraph on electronic effects) investigated NQEs in small organic molecules.<sup>161</sup> The study revealed that NQEs can dynamically strengthen molecular interactions by enhancing  $n \rightarrow \pi^*$  donation through increasing orbital overlap, or by strengthening electrostatic interactions between neighboring charge densities (see Figure 19C:3). Another interesting observed effect is a temporary change of bond orders, which can lead to emerging localized transient states of methyl rotors. The study also showed that vdW interactions are strengthened by NQEs: Since interatomic distances expand on average due to thermal and quantum dilations, the molecular polarizability is also increased (see Figure 19C:3). Other observed implications of NQEs include “bonding” between hydroxyl groups and hindered rotor dynamics, which leads to molecular stiffening and smoother Helmholtz free energy surfaces.

ML-FFs also make it possible to go far beyond the system size accessible with standard electronic structure methods. In ref 275, a descriptor-based NNP was used to study the influence of NQEs on aqueous NaOH solutions of different concentrations (~1000 atoms). It could be shown that NQEs exert a subtle influence on the solvation structure in the Na<sup>+</sup> environment and significantly increase the proton transfer rates and hence diffusion coefficients of the different species. The accuracy of the ML-FF also made it possible to identify error cancellation effects in the reference method, leading to

artificially good agreement with experiment in the absence of NQEs.

#### 4.5. Excited States

The Born–Oppenheimer approximation breaks down when modeling the dynamics of molecular excited states, which are essential for understanding photochemical processes. An extension to classical MD, which allows for the simulation of such phenomena, is quantum-classical surface hopping MD. In this approach, the excited state dynamics of a molecule are simulated by letting it evolve on a set of PESs associated with the different electronic states. To describe the distribution of the molecule between the different states, the effective PES governing the time evolution changes according to stochastic criteria, for example, based on coupling terms between the relevant states. The correct quantum statistics are then recovered from multiple independent simulations. These simulations are computationally intensive, as they not only require the computation of multiple PES but also different coupling terms. This is further amplified by the need for a large number of trajectories to obtain reliable statistics. As such, quantum-classical surface hopping simulations can profit greatly from the efficiency and versatility of ML-FFs.

In ref 276, for example, the authors used descriptor-based NNPs to study the excited state dynamics of the methylene imine molecule, as well as regions close to the conical intersection between the singlet ground and excited states. It could be shown that the NNPs are able to recover the effective PES with high accuracy and allow for efficient simulations to estimate the state populations of the system. Here, the coupling between the different surfaces was computed based on the Zhu–Nakamura approximation,<sup>301</sup> which relies on the energy differences between states. More accurate quantum mechanical descriptions of the interstate couplings rely on so-called nonadiabatic coupling vectors (NACs), which introduce several additional challenges from an ML perspective. First, NACs exhibit the same rotational equivariance as molecular forces. Second, they grow rapidly for states lying close in energy. And finally, as a quantity computed between different states, they are determined only up to an arbitrary phase. The latter property in particular complicates the construction of ML models, as the random nature of the phase factor needs to be compensated during training. Early works relied on a costly preprocessing of the reference data.<sup>277</sup> Ref 278, however, demonstrated that the phase problem can be overcome by introducing phase-less loss functions during the training procedure. Using a modified end-to-end NNP to describe the excited state dynamics of the methylenimmonium cation, it could be shown that using such loss terms completely eliminates the need for a preprocessing step. In addition, the work modeled the NACs as derivatives of a proxy potential, thus accounting for their transformation under rotations of the molecule. The combination of these approaches not only made it possible to obtain accurate population statistics for the studied system, but could also greatly extend the time scales accessible by the simulation beyond the limits of conventional electronic structure approaches.

For a recent review on machine learning for electronically excited states, see ref 302.

#### 4.6. Spectroscopy

As stated at the beginning of this section, MD simulations are excellent tools to model the temporal autocorrelation functions of various quantities, which can in turn be used to predict



experimental observables, such as diffusion coefficients. These quantities need not be restricted to properties derived from the PES, but they encompass other electronic properties such as dipole moments or polarizabilities. Access to the corresponding time autocorrelation functions enables the simulation of a wide range of molecular spectra, which can be directly related to experiment. The most prominent examples are infrared and Raman spectra derived from the autocorrelation functions of dipole moments and polarizabilities, respectively. Both types of vibrational spectra are of great practical interest since they can be measured accurately via experiment and provide insights into the atomic structure of molecules and materials. However, these spectra can be subject to a series of complex quantum mechanical effects such as vibrational anharmonicities. Hence, high level electronic structure treatments are required to obtain quantitatively accurate predictions of experimental results. Unfortunately, computing the required autocorrelation functions based purely on electronic structure calculations quickly grows prohibitively expensive, as simulations covering sufficient time scales are required to yield reliable spectra. In addition, if the influence of temperature or other phenomena should be studied in detail, a large number of such simulations is required. Recently, ML-FFs have emerged as invaluable tools for obtaining reliable molecular spectra. A growing number of ML-FFs now provide access to quantities beyond the PES, for example, dipole moments or polarizabilities. As such, they offer the possibility to perform these simulations in only a fraction of the time required by an *ab initio* approach or even make them possible at all.

Ref 71 demonstrates the potential inherent to ML-FFs based on the prediction of infrared spectra for organic molecules including the protonated alanine tripeptide. By combining a descriptor-based NNP model of the PES with a dipole moment model based on latent NN-predicted atomic charges, highly accurate infrared spectra could be obtained for all studied systems. The efficiency of such an approach was demonstrated based on an alkane containing more than 200 atoms, where it was possible to reduce a projected computation time of 9000 years with the original *ab initio* method to only a few days (including the reference calculations needed for training the ML models). Moreover, the high accuracy of the predictions made it possible to identify shortcomings of the original reference methods and study how they influence the infrared spectrum of the tripeptide. A similar latent charge based approach was employed in ref 195 to model infrared spectra of various amino acids. This study not only obtained accurate spectra but also demonstrated that the latent charges predicted by the dipole model constitute a valid ML driven scheme for deriving atomic partial charges, which can be used to model long-range electrostatic interactions explicitly. This scheme has since been employed in many physically augmented models (e.g., TensorMol<sup>195</sup> or PhysNet<sup>108</sup>).

In a similar manner, ML models capable of predicting polarizability tensors offer access to Raman spectra. Ref 279 introduces a symmetry adapted approach for modeling polarizability tensors using Gaussian process regression (GPR) based on the SOAP<sup>117</sup> kernel. The authors use this model to study the Raman spectra of paracetamol in gas phase and various molecular crystals and achieve excellent agreement with electronic structure methods in both cases. Not only is the proposed approach highly data efficient, requiring only a small number (<1000) of reference data, but it could also be shown that the resulting model is transferable between

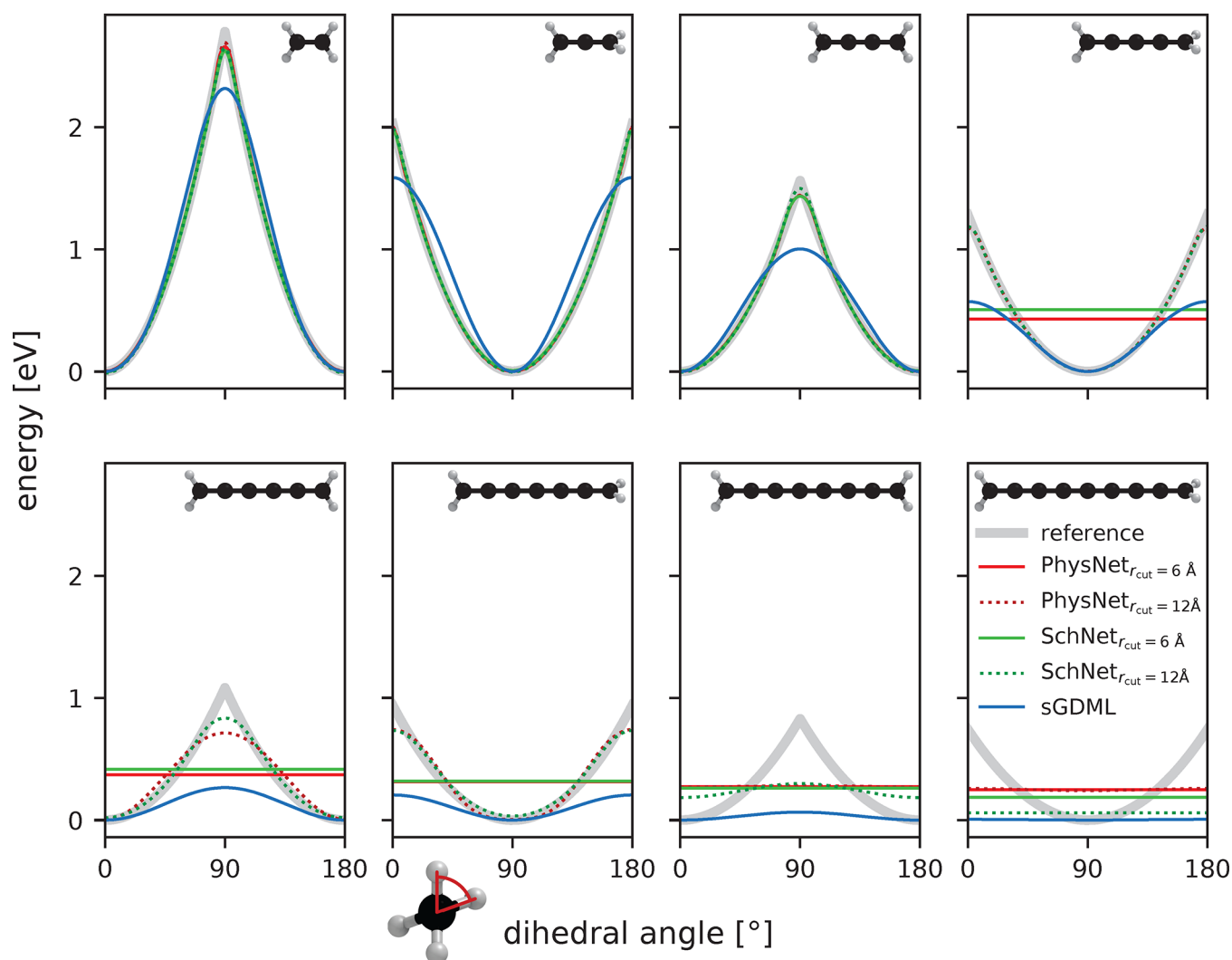
different polymorphic forms of the crystal. Ref 280 models Raman spectra of liquid water using descriptor-based NNPs to predict molecular polarizabilities. The computational efficiency of the approach made it possible to obtain Raman spectra for a system containing 416 water molecules based on two nanosecond trajectories at DFT level accuracy, a feat that would be infeasible with the original reference method. As a consequence, the influence of temperature effects on the Raman spectra of water and heavy water could be studied in detail. The atomic resolution of the employed ML approach made it possible to decompose the simulated spectra into intramolecular and intermolecular contributions, offering insights into the mechanisms governing the temperature dependence of the different spectral features.

## 5. CHALLENGES

Following the best practices outlined in the previous section, the current generation of ML-FFs is applicable to a wide range of problems in chemistry that involve small- to medium-sized systems. While this space of chemical compounds is already significant in size, the “dream scenario” of chemists and biologists referenced in the introduction can only be realized with access to larger system sizes. Not only does the number of stable structures increase exponentially with added atomistic degrees of freedom<sup>303,304</sup> but also many interesting phenomena play out at nanoscale resolution, which is inaccessible to ML methods as of yet. This is because some steps involved in the construction of ML-FFs, like sampling the reference data, which are solvable at small scale, become seemingly insurmountable obstacles at larger scales due to unfeasible computing times. The complexity of interactions, for example, the nonclassical behavior of nuclei, as well as significant contributions from large fluctuations, increase the space of conformations that need to be learned. To further complicate things, the cost of accurate *ab initio* calculations increases steeply with expanding system size, limiting the amount of reference data that can be collected within a reasonable time frame. This also means that a growing number of atom correlations need to be represented by a model to capture the full scope of interactions present in the real system. Below, some considerations in reconciling the somewhat contradicting demands of scalability, transferability, data efficiency and accuracy in large-scale ML-FFs are outlined.

### 5.1. Locality and Smoothness Assumptions

A fundamental challenge that must be faced by *ab initio* methods, conventional FFs, and ML models alike is the many-body problem. Most properties of a physical system are determined by the interaction of many particles, whether those are electrons or, on a higher abstraction level, atoms. In fact, the reason that *ab initio* calculations are expensive to obtain is due to the challenging computational scaling properties of high-dimensional many-body problems. As a result, the hierarchy of different levels of theory is directly defined by the level of correlation treatment in the respective wave function parametrization. Because the number of electronic degrees of freedom of a system is much higher than the number of atoms, the computational limitations of *ab initio* methods become evident very quickly, even for small systems. Atomistic approximations scale more favorably because they need to correlate less particles, but they are subject to the same scaling laws. The only escape is to neglect some correlations in favor of a reduced problem size. Unfortunately, it is to date



**Figure 20.** Energy profiles of different ML-based PESs for a rotation of the dihedral angle between the terminal methylene groups of cumulenes ( $C_{2+n}H_4$ ) of different sizes ( $0 \leq n \leq 7$ ). All reference calculations were performed with the semiempirical MNDO method<sup>309</sup> and models were trained on 4500 structures (with an additional 450 structures used for validation) collected from MD simulations at 1000 K. Because rotations of the dihedral angle are not sufficiently sampled at this temperature, the dihedral was rotated randomly before performing the reference calculations. Instead of a sharp cusp at the maximum of the rotation barrier, all models predict a smooth curve. Predictions become worse for increasing cumulenene sizes with the cusp region being oversmoothed more strongly. For  $n = 7$ , all models fail to predict the angular energy dependence. Note that NNP models (such as PhysNet and SchNet) may already fail for smaller cumulenes when the cutoff distance is chosen too small ( $r_{\text{cut}} = 6 \text{ \AA}$ ), as they are unable to encode information about the dihedral angle in the environment-descriptor. However, it is possible to increase the cutoff ( $r_{\text{cut}} = 12 \text{ \AA}$ ) to counter this effect.

impossible to reliably determine which interactions can be removed with minimal impact, without compromising the full many-body solution. Thus, the ideal of a local model is in conflict with the very nature of many-body systems. Although it is possible to recover some effects such as nonlocal charge transfer by means of a charge equilibration scheme,<sup>305,306</sup> a general solution for this problem does not exist. While not fully justified from a physics perspective, assuming locality is still a useful inductive bias, which can help generalization and computational efficiency. It also helps when collecting reference data, as it implies that larger systems can be predicted using the information learned from smaller systems. Another assumption, which all ML-FFs discussed in this review make, is that the PES is smooth. This is a necessary requirement for most practical applications since a nonsmooth PES implies force discontinuities, which would lead to instabilities during MD simulations. Smoothness is also a

requirement from the ML perspective, as only regular signals can be reconstructed from limited observations.

For most commonly used NNPs and many kernel-based ML-FFs, locality is built into the design explicitly through the introduction of a cutoff radius. The global interactions between atoms are modeled by accumulating individual local atomic contributions. In this “mean-field approximation”, the interaction of a particle with its surroundings is reduced to an effective one-body problem, that is, an interaction of that particle with the average effect of its neighbors. As similar neighborhoods can be identified in different compounds across chemical space, these assumptions allow to build models from reference calculations of small molecules, which are transferable to much larger structures.<sup>307,308</sup> However, the lack of explicit higher-order terms comes at the cost of potentially losing some important interaction effects, similar to the

Hartree–Fock method and Kohn–Sham DFT in *ab initio* calculations.

On the other hand, some models (e.g., (s)GDML) capture global correlations in the sense that a single prediction is obtained for the whole structure. Of course, this relies on reference calculations that are accurate enough to contain the relevant information. Global interactions of large systems can not be accurately inferred from a training set of small molecules or molecular fragments, which is why reference calculations for the exact target structure are necessary. It can therefore become difficult to collect enough reference data for large structures. In addition, even models that use no atom-wise decomposition might still implicitly assume that interactions are local to some degree due to their chemical descriptor. For example, in (s)GDML models, systems are encoded as a vector of inverse pairwise distances. Therefore, structural changes between distant atoms contribute less strongly to changes in the overall descriptor than proximal atoms.

While locality and smoothness are valid assumptions for the majority of chemical systems, there are pathological cases where they break down and ML models that rely on them perform poorly. As an example, consider cumulenes—hydrocarbons of the form  $C_{2+n}H_4$  ( $n \geq 0$ ) with  $n + 1$  cumulative double bonds. These molecules have a rigid linear geometry with the two terminal methylene groups forming an equilibrium dihedral angle of  $0^\circ$  (when  $n$  is even) or  $90^\circ$  (when  $n$  is odd). Rotating the dihedral angle out of its equilibrium position results in a sharp increase in potential energy even though the methylene groups may be separated by several angstroms when  $n$  is large. This is due to the energetically favorable overlap of  $\pi$ -orbitals along the carbon chain (a highly nonlocal interaction), which is broken when the methylene groups are rotated against each other. Additionally, the potential energy exhibits a sharp “cusp” at the maximum energy (i.e., it is not smooth), because the ground state electronic configuration switches abruptly from one state to another (strictly speaking, multireference calculations would be necessary here). One-dimensional projections of the PESs predicted by ML-FFs along the rotation of the dihedral angle reveal several problems (Figure 20). For example, all models predict smooth approximations by design, which is beneficial for running MD simulations, but results in large prediction errors around the cusp. Further, when the number of double bonds ( $n + 1$ ), that is, the “non-locality” of relevant interactions, is increased, the quality of predictions decreases dramatically, until all models are unable to reproduce the energy profile.

Note that by design, NNPs relying on message-passing are unable to resolve information about the dihedral angle if information between hydrogen atoms on opposite ends of the molecule cannot be exchanged directly (i.e.,  $r_{\text{cut}}$  is too small) and predict constant energies in this case. The same is true for descriptor-based NNPs, as fingerprints of chemical environments also only consider atoms up to a cutoff (see eqs 22 and 23). Any kernel method taking as input local structural descriptors relying on cutoff radii (e.g., SOAP<sup>117</sup> or FCHL19<sup>107</sup>) will suffer from the same problems. Even when a “global” descriptor of chemical structure such as inverse pairwise distances is chosen (e.g., Coulomb matrix<sup>28</sup>), changes in the dihedral angle between distant groups of atoms are not resolved sufficiently for accurate predictions (see sGDML model in Figure 20). The only way to fix this problem in

general is to drop the locality assumption completely, for example by including all  $\binom{N}{4}$  possible dihedral angles in the structural descriptor (without introducing additional factors that decrease the weight of these features with increasing distance between atoms). However, due to the combinatorial explosion of the number of possible dihedral angles, this would lead to extremely large descriptors whenever the number of atoms  $N$  is not very small. The resulting models would be slow to evaluate and require a lot of reference data to give robust predictions (to prevent them from entering the extrapolation regime). An expert choice, i.e., including only a single relevant dihedral angle in the descriptor, is a possible way around this issue, but requires prior knowledge of the problem at hand and goes somewhat against ML philosophy.

As a final remark, it should be mentioned that conventional FFs only include terms for dihedral angles between directly bonded atoms, so they are equally unable to predict the energy profiles of the larger cumulenes shown in Figure 20. As such, relying on chemical locality is an assumption made by virtually all methods for approximating PESs and is not specific to just ML methods.

## 5.2. Transferability, Scalability, and Long-Range Interactions

The concept of chemical locality discussed above also plays a central role in the transferability and scalability of ML models for atomistic systems. Transferability indicates how well models can adapt to compounds varying in their chemical composition, while scalability indicates how efficiently these models scale with respect to the size of systems modeled. Both concepts are closely related and inherently rooted in chemical locality. The assumption that interactions between atoms are local implies that similar structural motifs will give rise to comparable interactions and hence similar contributions to the properties of a molecule or material. In an ML context, chemical locality allows a model to reuse the information learned for different parts of a molecule for similar features in different systems. In this manner, a large atomistic system could in principle be assembled from smaller components like a jigsaw puzzle.<sup>307</sup> The former aspect is crucial to make models transferable, while the latter allows for the development of architectures whose evaluation cost scales linearly with system size.

ML-FFs exploiting chemical locality offer several advantages compared to other models. If trained properly, they can be applied to systems of different size and composition. The training procedure benefits in a similar manner, as local models can be trained on structures containing different numbers of atoms. Moreover, it is also possible to use only fragments of the original system during construction of a model. This property is very attractive in situations where accurate reference computations for the whole system are infeasible due to system size or scaling of the computational method. Local chemical environments are also less diverse than complete structures, potentially reducing the need for extensive sampling and decreasing the chances that models enter the extrapolation regime in a production setting. In addition, local models scale linearly with system size, as interactions are limited to the cutoff radius and can be evaluated efficiently. In contrast, models without cutoffs are typically more limited in their practical applicability for extended systems. They always require reference computations to be performed for the whole

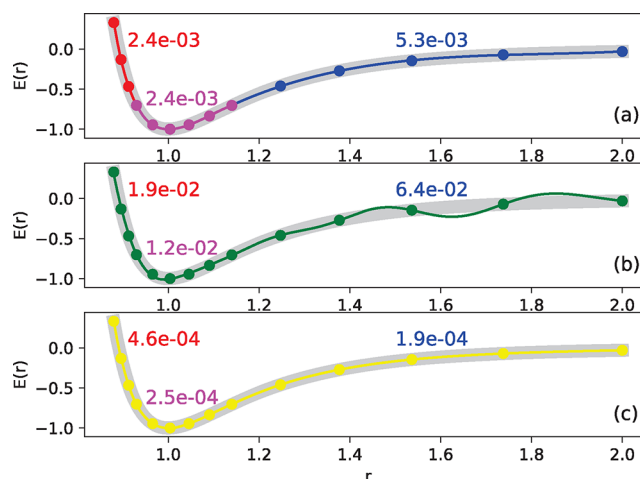


system and, once trained, can only be reused for this particular molecule or material.

Despite these advantages, local ML models suffer from several inherent problems. To construct models that exploit locality, a chemical system needs to be partitioned in one way or another. This can for example be achieved by limiting interactions to terms involving only a certain number of atoms (similar to conventional FFs) or by restricting them to local atom-centered environments. These approximations place strong limitations on which kind of interactions can be described. As a result, local ML models have difficulty when dealing with the situations where nonlocal effects are important, such as strongly conjugated systems and excited states (see Section 5.1). For standard simulations, the presence of long-range interactions, such as electrostatic and dispersion effects, are much more common phenomena. These are particularly important for modeling extended systems, where ML models are typically believed to offer a significant advantage over more conventional FFs. Since the structure and dynamical behavior of such systems is influenced greatly by long-range interactions, ML models need to be able to account for them in a satisfying manner.

Recovering long-range effects necessitates a balancing act between physical accuracy and computational efficiency, as the scalability of local models hinges on there being a limited number of interactions which need to be evaluated. This feat is further complicated by the typical energy scales of these interactions, which are small compared to local contributions such as bond energies. For these reasons, it is not advisable to account for long-range interactions by simply increasing the size of local environments. While local models with sufficiently large cutoffs are able to learn the relevant effects in principle, it may require a disproportionately large amount of data to reach an acceptable level of accuracy for an interaction with a comparably simple functional form. The reason is that average gradients and curvature in different regions of the PES may differ by several orders of magnitude, which makes it difficult to achieve uniformly low prediction errors across all regions. Hence, an optimal description would require to employ different characteristic scales.

For illustration, consider the following toy examples: In the first variant, a Lennard-Jones (LJ) potential<sup>14</sup> is separated into a region around its minimum, a repulsive short-range, and an attractive long-range part. The task is to learn each of the three regions with a separate model (see Figure 21a). In the second variant, a single model is trained on all regions at once (see Figure 21b). Here, all models are kernel-based and use a Gaussian kernel (eq 4). The kernel hyper-parameter  $\gamma$  is optimized by a grid-search and cross-validation. Compared to the models trained on individual regions, the prediction errors of the model for all regions increase by around an order of magnitude. Further, it shows spurious oscillations between training points in the long-range region. When the optimal values of  $\gamma$  for the different models are compared, the reason for failure when training on all regions at once becomes apparent: The optimal values of  $\gamma$  are 198.88, 75.47, 0.08 for the short-, middle-, and long-range models, respectively, which highlights the multiscale nature of the PES. On the other hand, when training on all regions at once, the model necessarily has to compromise, which leads to an optimal value of  $\gamma = 22.12$ . In this toy example, the multiscale problem can be solved by switching from using  $r$  as a structural descriptor to the more appropriate inverse distance  $r^{-1}$  (Figure 21c). Unfortunately,



**Figure 21.** Lennard-Jones potential (thick gray line) predicted by KRR with a Gaussian kernel. (a) Potential energy is decomposed into short- (red), middle- (magenta), and long-range (blue) parts, which are learned by separate models (symbols show the training data and solid lines the model predictions). The mean squared prediction errors (in arbitrary units) for the respective regions are shown in the corresponding colors. (b) Entire potential is learned by a single model using the same training points (green). All models in panels a and b use  $r$  as the structural descriptor. (c) Single model learning the potential, but using  $r^{-1}$  as structural descriptor (yellow). The mean squared errors (a.u.) for different parts of the potential in panels b and c are reported independently to allow direct comparison with the values reported in panel a.

for realistic (high-dimensional) PESs with multiple minima, it can be difficult to find an appropriate descriptor to address the multiscale nature of the PES, which leads to data-inefficient models. As a result, more training data is needed to reach an acceptable accuracy, which is problematic considering the computational cost of high-quality reference calculations.

One possibility of overcoming these limitations is by instead partitioning the energy into contributions modeled entirely via ML (short-range) and contributions described via explicit physical relations based on local quantities predicted via ML (long-range). A prime example for such an approach is the treatment of electrostatics, as was first introduced in ref 221. Here, an ML model is used to predict partial charges for each atom based on their local environment. These charges can then be used in standard Coulomb and Ewald summation to compute the long-range electrostatic energy of a system. While such schemes initially relied on point charge reference data obtained from (arbitrary) partitioning methods of the *ab initio* electron density (e.g., Hirshfeld charges<sup>310</sup>), they have since been extended to operate on charges derived from an ML model for dipole moments (a true quantum mechanical observable).<sup>71,108,195</sup> Here, scalar partial charges  $q_i$  are predicted for each atom  $i$  and the molecular dipole moment is constructed as  $\mu = \sum_i q_i r_i$ , where  $r_i$  are the atomic positions (the predicted  $q_i$  can be corrected to guarantee charge conservation<sup>108</sup>). The discrepancy between reference and predicted dipole moments is included in the loss function used for training the model (see Section 3.5) and the partial charges consequently derived in a purely data-driven manner.

Contrary to electrostatics, accounting for dispersion interactions is not as straightforward, because the exact physical form of dispersion interactions is still debated and a variety of approximate schemes have been proposed.<sup>281</sup> In

addition, dispersion corrections typically depend on coefficients computed from atomic polarizabilities as local properties. The corresponding quantum mechanical observable is the molecular polarizability tensor. In contrast to charges (scalars) derived from dipole moments (a vector quantity), predicting molecular polarizabilities requires rotationally equivariant ML models.<sup>311</sup> Because of this, many ML approaches rely on the same empirical pairwise dispersion potentials employed for correcting density functional theory computations.<sup>108,312,313</sup>

To summarize, local ML architectures are a promising approach toward transferable and scalable models, but they have a number of drawbacks that will still need to be addressed in the future. Promising alternative approaches to achieve transferability are ML models based directly on electronic structure methods, i.e., “semi-empirical ML”<sup>57,314,315</sup> and models for electron density and Hamiltonians.<sup>55</sup> These approaches express fundamental quantum chemical quantities in a local representation, for example, Hamiltonian matrix elements in an atomic orbital basis. Nonlocality can then be introduced via the “correct” mathematical mechanism, for example, matrix diagonalization in the case of Hamiltonians. This physically motivated structure allows such models to recover a wide range of interactions while still being transferable. They are also better suited to predict intensive properties of molecules (whose magnitude is independent of system size), where assuming additive atomic contributions is not valid. A downside of such models compared to conventional ML-FFs is the increased computational cost due to the additional matrix operations.

With respect to scalability, hybrid approaches similar to QM/MM<sup>25</sup> might constitute valid alternatives to pure ML models. Although several orders of magnitude more efficient than electronic structure theory, even local ML models encounter problems when faced with systems containing tens of thousands of atoms. Compared to conventional FFs, the more complex functional form underlying ML-FFs leads to an increased computational cost. In such cases, partitioning the system into regions treated at different levels of approximation can lead to a significant speedup. ML models can for example be embedded into regions modeled by classical force fields, yielding ML/MM like simulation protocols. Restricting elaborate ML approaches to only a subset of a chemical systems would make it possible to employ more accurate approximations in a manner analogous to conventional QM/MM. For example, in ref 316, the authors study protein–ligand binding with a ML/MM approach. The ligand is described by an NN-based ML-FF and treated as if it was in gas phase. Coupling to the protein environment (described by a conventional FF) is achieved solely through nonbonded dispersion and electrostatic interactions. The disadvantage of such a simple embedding is that the “quantum region” cannot be polarized by the “classical region”. A more sophisticated embedding was recently proposed by Gastegger et al.<sup>317</sup> Here, the region described by the ML-FF is explicitly polarized by the electric field induced by surrounding point charges, that is, the electric field is an additional input to the model. Alternative approaches, describe the effect of the classical environment by augmenting structural descriptors such as ACSFs by additional terms explicitly depending on the MM point charges.<sup>318</sup> A similar approach is followed in ref 319, where the classical environment is described by auxiliary atom types.

## 6. CONCLUDING REMARKS

The last decades have witnessed significant advances in statistical learning that allowed ML techniques to enter our daily lives, industrial practice, and scientific research.

Classically, automation in industry and scientific fields relied on hand-crafted rules that represented human knowledge.<sup>320</sup> Not only is the creation of rule-based systems laborious and may require to cover an excessive number of cases, it often leads to rigid structures that are unable to adapt well to new situations. Even worse, some concepts are difficult or impossible to formalize, such as human perception for image classification.

Modern statistical ML algorithms<sup>98,128</sup> such as deep learning<sup>94,321–323</sup> or kernel-based learning<sup>90,92,126,324,325</sup> enable models that freely adapt to knowledge that is implicitly contained in data sets (in an abstract form) and thus offer a more robust way of solving problems than rule-based reasoning. For the field of molecular simulations, the potential of ML methods may help to bridge the accuracy-efficiency gap between first-principles electronic structure methods and conventional (rule-based) FFs. Bringing both fields together has raised many questions and still poses some fundamental challenges for new generations of ML-FFs. At this point in time, ML-FFs have already become a successful and practical tool in computational chemistry.

Starting from a broad perspective, this review has focused on the role of ML for constructing force fields and assessed what can be achieved with these new techniques at the current stage of development. This has been contrasted with problems that are (so far) beyond the reach of present methods. Illustrative examples of the relevant chemistry and ML concepts have been discussed to demonstrate the practical usefulness that modern ML techniques can bring to chemistry and physics. This includes an overview of the most important considerations behind the construction of modern ML-FFs such as the incorporation of physical invariances, choice of ML algorithms, and loss functions. Special attention has been given to the topic of validating ML-FFs, which requires particular care in scientific applications.<sup>326</sup> Furthermore, a comprehensive list of best practices, pitfalls, and challenges has been provided, which will serve as a useful guideline for practitioners standing on either side of this growing interdisciplinary field. These “tricks of the trade”<sup>157</sup> are often assumed to be obvious and thus omitted from publications—here they have been deliberately spelled out to avoid unnecessary barriers to enter the field. Additionally, a small catalog of software tools that can enable and accelerate the implementation of ML-FFs has been provided as a pointer for readers wishing to adopt ML methods in their own research.

While routinely performing computational studies of condensed phase systems (e.g., proteins in solution) at the highest levels of theory is still beyond reach, ML methods have already made other “smaller dreams” a reality. Just a decade ago, it would have been unthinkable to study the dynamics of molecules like aspirin at coupled cluster accuracy. Today, a couple hundred *ab initio* reference calculations are enough to construct ML-FFs that reach this accuracy within a few tens of wavenumbers.<sup>327</sup> In the past, even if suitable reference data was available, constructing accurate force fields was labor-intensive and required human effort and expertise. Nowadays, by virtue of automatic ML methods, the same task is as effortless as the push of a button. Thanks to the speed-ups

offered by ML methods over conventional approaches, studies that previously required supercomputers to be feasible in a realistic time frame<sup>287,289</sup> can now be performed on a laptop computer.<sup>69,109</sup>

In addition to enabling studies that were prohibitively expensive in the past, ML methods have also led to new chemical insights on systems that were thought to be already well understood. Even relatively small molecules were shown to display nontrivial electronic effects, influencing their dynamics and allowing a better understanding of experimental observations.<sup>161</sup> Many other unknown chemical effects potentially wait to be discovered by studies now possible with ML-FFs. At the speed at which improvements to existing ML-FFs are published, it is not unreasonable to expect significant advances that will make similar studies possible for larger systems and help realize many more “dreams” in the near future.

Concluding, ML-FFs are a highly active line of research with many unexplored avenues and attractive applications in chemistry, with possibilities to contribute to a better understanding of fundamental quantum chemical properties and ample opportunity for novel theoretical, algorithmic, and practical improvement. Given the success of this relatively young interdisciplinary field, it is to be expected that ML-FFs will become a fundamental part of modern computational chemistry.

## AUTHOR INFORMATION

### Corresponding Authors

**Alexandre Tkatchenko** – Department of Physics and Materials Science, University of Luxembourg, L-1511 Luxembourg City, Luxembourg; [orcid.org/0000-0002-1012-4854](https://orcid.org/0000-0002-1012-4854); Email: [alexandre.tkatchenko@uni.lu](mailto:alexandre.tkatchenko@uni.lu)

**Klaus-Robert Müller** – Machine Learning Group, Technische Universität Berlin, 10587 Berlin, Germany; BIFOLD–Berlin Institute for the Foundations of Learning and Data, Berlin, Germany; Department of Artificial Intelligence, Korea University, Seongbuk-gu, Seoul 02841, Korea; Max Planck Institute for Informatics, Stuhlsatzenhausweg, 66123 Saarbrücken, Germany; Google Research, Brain Team, Berlin, Germany; [orcid.org/0000-0002-3861-7685](https://orcid.org/0000-0002-3861-7685); Email: [klaus-robert.mueller@tu-berlin.de](mailto:klaus-robert.mueller@tu-berlin.de)

### Authors

**Oliver T. Unke** – Machine Learning Group, Technische Universität Berlin, 10587 Berlin, Germany; DFG Cluster of Excellence “Unifying Systems in Catalysis” (UniSysCat), Technische Universität Berlin, 10623 Berlin, Germany; [orcid.org/0000-0001-7503-406X](https://orcid.org/0000-0001-7503-406X)

**Stefan Chmiela** – Machine Learning Group, Technische Universität Berlin, 10587 Berlin, Germany

**Huziel E. Saucedo** – Machine Learning Group and BASLEARN, BASF-TU Joint Lab, Technische Universität Berlin, 10587 Berlin, Germany; [orcid.org/0000-0001-6091-3408](https://orcid.org/0000-0001-6091-3408)

**Michael Gastegger** – Machine Learning Group and BASLEARN, BASF-TU Joint Lab, Technische Universität Berlin, 10587 Berlin, Germany; DFG Cluster of Excellence “Unifying Systems in Catalysis” (UniSysCat), Technische Universität Berlin, 10623 Berlin, Germany

**Igor Poltavsky** – Department of Physics and Materials Science, University of Luxembourg, L-1511 Luxembourg City, Luxembourg; [orcid.org/0000-0002-3188-7017](https://orcid.org/0000-0002-3188-7017)

**Kristof T. Schütt** – Machine Learning Group, Technische Universität Berlin, 10587 Berlin, Germany; [orcid.org/0000-0001-8342-0964](https://orcid.org/0000-0001-8342-0964)

Complete contact information is available at:  
<https://pubs.acs.org/10.1021/acs.chemrev.0c01111>

### Author Contributions

△O.T.U. and S.C. contributed equally.

### Notes

The authors declare no competing financial interest.

### Biographies

Oliver T. Unke is an SNSF postdoctoral research fellow in the Machine Learning Group at Technische Universität Berlin. He received his Ph.D. in Chemistry from the University of Basel in 2019. His research has focused on developing methods for constructing accurate potential energy surfaces and their application in molecular dynamics simulations.

Stefan Chmiela is a senior researcher at the Berlin Institute for the Foundations of Learning and Data (BIFOLD). He received his Ph.D. from Technische Universität Berlin in 2019. His research interests include Hilbert space learning methods for applications in quantum chemistry, with particular focus on data efficiency and robustness.

Huziel E. Saucedo is a postdoctoral researcher in the Machine Learning Group at Technische Universität Berlin and part of the BASLEARN project at the same institution. He obtained his bachelor degree in Physics at the Universidad Autónoma de Sinaloa in Mexico, and his master and Ph.D. at the Institute of Physics of the Universidad Nacional Autónoma de México (UNAM) in Mexico City. Between 2016 and 2019, he was a postdoctoral researcher at the Fritz Haber Institute of the Max Planck Society in Berlin. His research interests include *ab initio* simulations, nuclear quantum effects, and thermodynamics of (nano)materials, as well as development and applications of machine learning methods to quantum chemistry and materials science.

Michael Gastegger is a postdoctoral researcher in the BASLEARN project of the Machine Learning Group at Technische Universität Berlin. He received his Ph.D. in Chemistry from the University of Vienna in Austria in 2017. His research interests include the development of machine learning methods for quantum chemistry and their application in simulations.

Igor Poltavsky is a senior researcher at the University of Luxembourg. He received his Ph.D. from B. Verkin Institute for Low Temperature Physics & Engineering in 2009. His research interests include statistical physics, imaginary-time path integral methods, nuclear quantum effects, *ab initio* simulations, and machine learning.

Kristof T. Schütt is a senior researcher at the Berlin Institute for the Foundations of Learning and Data (BIFOLD). He received his master's degree in computer science in 2012 and his Ph.D. at the machine learning group of Technische Universität Berlin in 2018. Until September 2020, he worked at the Audatic company developing neural networks for real-time speech enhancement. His research interests include interpretable neural networks, representation learning, generative models, and machine learning applications in quantum chemistry.

Alexandre Tkatchenko is a Professor of Theoretical Chemical Physics at the University of Luxembourg and Visiting Professor at Technische Universität Berlin. He obtained his bachelor degree in Computer Science and a Ph.D. in Physical Chemistry at the Universidad Autónoma Metropolitana in Mexico City. Between 2008 and 2010, he



was an Alexander von Humboldt Fellow at the Fritz Haber Institute of the Max Planck Society in Berlin. Between 2011 and 2016, he led an independent research group at the same institute. Tkatchenko serves on editorial boards of two society journals: Physical Review Letters (APS) and Science Advances (AAAS). He received a number of awards, including elected Fellow of the American Physical Society, the 2020 Dirac Medal from WATOC, the Gerhard Ertl Young Investigator Award of the German Physical Society, and two flagship grants from the European Research Council: a Starting Grant in 2011 and a Consolidator Grant in 2017. His group pushes the boundaries of quantum mechanics, statistical mechanics, and machine learning to develop efficient methods to enable accurate modeling and obtain new insights into complex materials.

Klaus-Robert Müller studied physics at the Technische Universität Karlsruhe, Karlsruhe, Germany, from 1984 to 1989. He received his Ph.D. degree in computer science from Technische Universität Karlsruhe in 1992. He has been a Professor of computer science with Technische Universität Berlin (TU Berlin), Berlin, Germany, since 2006. In 2020 and 2021, he is on a sabbatical leave from TU Berlin and with the Brain Team, Google Research, Berlin, Germany. He is also directing and codirecting the Berlin Machine Learning Center, Berlin, and the Berlin Big Data Center, Berlin, respectively. After completing a postdoctoral position at GMD FIRST, Berlin, he was a Research Fellow with The University of Tokyo, Tokyo, Japan, from 1994 to 1995. In 1995, he founded the Intelligent Data Analysis Group at GMD FIRST (later Fraunhofer FIRST) and directed it until 2008. From 1999 to 2006, he was a Professor with the University of Potsdam, Potsdam, Germany. His research interests are intelligent data analysis and machine learning in the sciences (neuroscience (specifically brain–computer interfaces), physics, and chemistry) and in industrial applications. Dr. Müller was elected member of the German National Academy of Sciences, Leopoldina, in 2012 and the Berlin Brandenburg Academy of Sciences in 2017 and an External Scientific Member of the Max Planck Society in 2017. In 2019 and 2020, he became a Highly Cited Researcher in the cross-disciplinary area. Among others, he was awarded the Olympus Prize for Pattern Recognition in 1999, the SEL Alcatel Communication Award in 2006, the Science Prize of Berlin by the Governing Mayor of Berlin in 2014, the Vodafone Innovations Award in 2017, and the 2020 Best Paper Award of the journal Pattern Recognition.

## ACKNOWLEDGMENTS

O.T.U. acknowledges funding from the Swiss National Science Foundation (Grant No. P2BSP2\_188147). A.T. was supported by the European Research Council (ERC-CoG “BeStMo”). K.-R.M. was supported in part by the Institute of Information & Communications Technology Planning & Evaluation (IITP) grants funded by the Korea Government (No. 2017-0-00451, Development of BCI based Brain and Cognitive Computing Technology for Recognizing User’s Intentions using Deep Learning) and funded by the Korea Government (No. 2019-0-00079, Artificial Intelligence Graduate School Program, Korea University), and was partly supported by the German Ministry for Education and Research (BMBF) under Grant Nos. 01IS14013A-E, 01GQ1115, 01GQ0850, 01IS18025A, 03IL0207D, and 01IS18037A; the German Research Foundation (DFG) under Grant Math+, EXC 2046/1, Project ID 390685689. We would like to thank Stefan Ganschä for the valuable input to the manuscript.

## REFERENCES

- (1) Feynman, R. P.; Leighton, R. B.; Sands, M. *The Feynman Lectures On Physics*; Addison-Wesley, 1963; Vol. 1.
- (2) McCammon, J. A.; Gelin, B. R.; Karplus, M. Dynamics Of Folded Proteins. *Nature* **1977**, *267*, 585–590.
- (3) Phillips, D. *Biomolecular Stereodynamics*; Adenine Press: Gunderland, NY, 1981.
- (4) Schulz, R.; Lindner, B.; Petridis, L.; Smith, J. C. Scaling Of Multimillion-atom Biological Molecular Dynamics Simulation On A Petascale Supercomputer. *J. Chem. Theory Comput.* **2009**, *5*, 2798–2808.
- (5) Shaw, D. E.; Deneroff, M. M.; Dror, R. O.; Kuskin, J. S.; Larson, R. H.; Salmon, J. K.; Young, C.; Batson, B.; Bowers, K. J.; Chao, J. C.; et al. Anton, A Special-purpose Machine For Molecular Dynamics Simulation. *Commun. ACM* **2008**, *51*, 91–97.
- (6) Freddolino, P. L.; Arkhipov, A. S.; Larson, S. B.; McPherson, A.; Schulten, K. Molecular Dynamics Simulations Of The Complete Satellite Tobacco Mosaic Virus. *Structure* **2006**, *14*, 437–449.
- (7) Zhao, G.; Perilla, J. R.; Yufenyuy, E. L.; Meng, X.; Chen, B.; Ning, J.; Ahn, J.; Gronenborn, A. M.; Schulten, K.; Aiken, C.; et al. Mature HIV-1 Capsid Structure By Cryo-electron Microscopy And All-atom Molecular Dynamics. *Nature* **2013**, *497*, 643–646.
- (8) Zimmerman, M. I.; Porter, J. R.; Ward, M. D.; Singh, S.; Vithani, N.; Meller, A.; Mallimadugula, U. L.; Kuhn, C. E.; Borowsky, J. H.; Wiewiora, R. P., et al. Citizen Scientists Create An Exascale Computer To Combat COVID-19. *bioRxiv* **2020**, DOI: 10.1101/2020.06.27.175430.
- (9) Karplus, M.; McCammon, J. A. Molecular Dynamics Simulations Of Biomolecules. *Nat. Struct. Biol.* **2002**, *9*, 646–652.
- (10) Dirac, P. A. M. Quantum Mechanics Of Many-electron Systems. *Proc. R. Soc. London A* **1929**, *123*, 714–733.
- (11) Gordon, M. S.; Schmidt, M. W. Advances in electronic structure theory. *Theory and Applications of Computational Chemistry* **2005**, 1167–1189.
- (12) González, M. Force Fields And Molecular Dynamics Simulations. *École thématique de la Société Française de la Neutronique* **2011**, *12*, 169–200.
- (13) Unke, O. T.; Koner, D.; Patra, S.; Käser, S.; Meuwly, M. High-dimensional Potential Energy Surfaces For Molecular Simulations: From Empiricism To Machine Learning. *Mach. Learn.: Sci. Technol.* **2020**, *1*, 13001.
- (14) Lennard-Jones, J. E. On The Determination Of Molecular Fields. – II. From The Equation Of State Of A Gas. *Proc. R. Soc. London A* **1924**, *106*, 463–477.
- (15) Vitalini, F.; Mey, A. S.; Noé, F.; Keller, B. G. Dynamic Properties Of Force Fields. *J. Chem. Phys.* **2015**, *142*, 02B611\_1.
- (16) Halgren, T. A.; Damm, W. Polarizable Force Fields. *Curr. Opin. Struct. Biol.* **2001**, *11*, 236–242.
- (17) Warshel, A.; Kato, M.; Pisiakov, A. V. Polarizable Force Fields: History, Test Cases, And Prospects. *J. Chem. Theory Comput.* **2007**, *3*, 2034–2045.
- (18) Shi, Y.; Xia, Z.; Zhang, J.; Best, R.; Wu, C.; Ponder, J. W.; Ren, P. Polarizable Atomic Multipole-based AMOEBA Force Field For Proteins. *J. Chem. Theory Comput.* **2013**, *9*, 4046–4063.
- (19) Lopes, P. E.; Huang, J.; Shim, J.; Luo, Y.; Li, H.; Roux, B.; MacKerell, A. D., Jr Polarizable Force Field For Peptides And Proteins Based On The Classical Drude Oscillator. *J. Chem. Theory Comput.* **2013**, *9*, 5430–5449.
- (20) Rasmussen, T. D.; Ren, P.; Ponder, J. W.; Jensen, F. Force Field Modeling Of Conformational Energies: Importance Of Multipole Moments And Intramolecular Polarization. *Int. J. Quantum Chem.* **2007**, *107*, 1390–1395.
- (21) Unke, O. T.; Devereux, M.; Meuwly, M. Minimal Distributed Charges: Multipolar Quality At The Cost Of Point Charge Electrostatics. *J. Chem. Phys.* **2017**, *147*, 161712.
- (22) Warshel, A.; Weiss, R. M. An Empirical Valence Bond Approach For Comparing Reactions In Solutions And In Enzymes. *J. Am. Chem. Soc.* **1980**, *102*, 6218–6226.

- (23) Van Duin, A. C.; Dasgupta, S.; Lorant, F.; Goddard, W. A. ReaxFF: A Reactive Force Field For Hydrocarbons. *J. Phys. Chem. A* **2001**, *105*, 9396–9409.
- (24) Nagy, T.; Yosa Reyes, J.; Meuwly, M. Multisurface Adiabatic Reactive Molecular Dynamics. *J. Chem. Theory Comput.* **2014**, *10*, 1366–1375.
- (25) Senn, H. M.; Thiel, W. QM/MM Methods For Biomolecular Systems. *Angew. Chem., Int. Ed.* **2009**, *48*, 1198–1229.
- (26) Kulik, H. J.; Zhang, J.; Klinman, J. P.; Martinez, T. J. How Large Should The QM Region Be In QM/MM Calculations? The Case Of Catechol O-Methyltransferase. *J. Phys. Chem. B* **2016**, *120*, 11381–11394.
- (27) Schütt, K. T.; Chmiela, S.; von Lilienfeld, O. A.; Tkatchenko, A.; Tsuda, K.; Müller, K.-R. *Machine Learning Meets Quantum Physics*; Springer, 2020.
- (28) Rupp, M.; Tkatchenko, A.; Müller, K.-R.; Von Lilienfeld, O. A. Fast and Accurate Modeling of Molecular Atomization Energies with Machine Learning. *Phys. Rev. Lett.* **2012**, *108*, 058301.
- (29) Montavon, G.; Rupp, M.; Gobre, V.; Vazquez-Mayagoitia, A.; Hansen, K.; Tkatchenko, A.; Müller, K.-R.; Von Lilienfeld, O. A. Machine Learning of Molecular Electronic Properties in Chemical Compound Space. *New J. Phys.* **2013**, *15*, 095003.
- (30) Hansen, K.; Montavon, G.; Biegler, F.; Fazli, S.; Rupp, M.; Scheffler, M.; Von Lilienfeld, O. A.; Tkatchenko, A.; Müller, K.-R. Assessment and Validation of Machine Learning Methods for Predicting Molecular Atomization Energies. *J. Chem. Theory Comput.* **2013**, *9*, 3404–3419.
- (31) Hansen, K.; Biegler, F.; Ramakrishnan, R.; Pronobis, W.; Von Lilienfeld, O. A.; Müller, K.-R.; Tkatchenko, A. Machine Learning Predictions Of Molecular Properties: Accurate Many-body Potentials And Nonlocality In Chemical Space. *J. Phys. Chem. Lett.* **2015**, *6*, 2326–2331.
- (32) von Lilienfeld, O. A.; Müller, K.-R.; Tkatchenko, A. Exploring Chemical Compound Space With Quantum-based Machine Learning. *Nat. Rev. Chem.* **2020**, *4*, 347–358.
- (33) Noé, F.; Olsson, S.; Köhler, J.; Wu, H. Boltzmann Generators: Sampling Equilibrium States Of Many-body Systems With Deep Learning. *Science* **2019**, *365*, eaaw1147.
- (34) Köhler, J.; Klein, L.; Noé, F. Equivariant Flows: Exact Likelihood Generative Learning For Symmetric Densities. *Proceedings of the 37th International Conference on Machine Learning* **2020**, *119*, 5361–5370.
- (35) Zhang, J.; Yang, Y. I.; Noé, F. Targeted Adversarial Learning Optimized Sampling. *J. Phys. Chem. Lett.* **2019**, *10*, 5791–5797.
- (36) Koner, D.; Unke, O. T.; Boe, K.; Bemish, R. J.; Meuwly, M. Exhaustive State-to-state Cross Sections For Reactive Molecular Collisions From Importance Sampling Simulation And A Neural Network Representation. *J. Chem. Phys.* **2019**, *150*, 211101.
- (37) Noé, F.; De Fabritiis, G.; Clementi, C. Machine Learning For Protein Folding And Dynamics. *Curr. Opin. Struct. Biol.* **2020**, *60*, 77–84.
- (38) Sønderby, S. K.; Sønderby, C. K.; Nielsen, H.; Winther, O. Convolutional LSTM Networks For Subcellular Localization Of Proteins. *International Conference on Algorithms for Computational Biology*. **2015**, 9199, 68–80.
- (39) Almagro Armenteros, J. J.; Sønderby, C. K.; Sønderby, S. K.; Nielsen, H.; Winther, O. DeepLoc: Prediction Of Protein Subcellular Localization Using Deep Learning. *Bioinformatics* **2017**, *33*, 3387–3395.
- (40) Botlani, M.; Siddiqui, A.; Varma, S. Machine Learning Approaches To Evaluate Correlation Patterns In Allosteric Signaling: A Case Study Of The PDZ2 Domain. *J. Chem. Phys.* **2018**, *148*, 241726.
- (41) Boninsegna, L.; Nüske, F.; Clementi, C. Sparse Learning Of Stochastic Dynamical Equations. *J. Chem. Phys.* **2018**, *148*, 241723.
- (42) Senior, A. W.; Evans, R.; Jumper, J.; Kirkpatrick, J.; Sifre, L.; Green, T.; Qin, C.; Židek, A.; Nelson, A. W.; Bridgland, A.; et al. Improved Protein Structure Prediction Using Potentials From Deep Learning. *Nature* **2020**, *577*, 706–710.
- (43) Scherer, M. K.; Trendelkamp-Schroer, B.; Paul, F.; Pérez-Hernández, G.; Hoffmann, M.; Plattner, N.; Wehmeyer, C.; Prinz, J.-H.; Noé, F. PyEMMA 2: A Software Package For Estimation, Validation, And Analysis Of Markov Models. *J. Chem. Theory Comput.* **2015**, *11*, 5525–5542.
- (44) Mardt, A.; Pasquali, L.; Wu, H.; Noé, F. VAMPnets For Deep Learning Of Molecular Kinetics. *Nat. Commun.* **2018**, *9*, 1–11.
- (45) Wehmeyer, C.; Noé, F. Time-lagged Autoencoders: Deep Learning Of Slow Collective Variables For Molecular Kinetics. *J. Chem. Phys.* **2018**, *148*, 241703.
- (46) Wu, H.; Mardt, A.; Pasquali, L.; Noe, F. Deep Generative Markov State Models. *Adv. Neural. Inf. Process. Syst.* **2018**, 3975–3984.
- (47) Chen, W.; Sidky, H.; Ferguson, A. L. Nonlinear Discovery Of Slow Molecular Modes Using State-free Reversible VAMPnets. *J. Chem. Phys.* **2019**, *150*, 214114.
- (48) Klus, S.; Husic, B. E.; Mollenhauer, M.; Noé, F. Kernel Methods For Detecting Coherent Structures In Dynamical Data. *Chaos* **2019**, *29*, 123112.
- (49) Sidky, H.; Chen, W.; Ferguson, A. L. High-resolution Markov State Models For The Dynamics Of Trp-cage Miniprotein Constructed Over Slow Folding Modes Identified By State-free Reversible VAMPnets. *J. Phys. Chem. B* **2019**, *123*, 7999–8009.
- (50) Chen, J.; Chen, J.; Pinamonti, G.; Clementi, C. Learning Effective Molecular Models From Experimental Observables. *J. Chem. Theory Comput.* **2018**, *14*, 3849–3858.
- (51) Wang, J.; Olsson, S.; Wehmeyer, C.; Pérez, A.; Charron, N. E.; De Fabritiis, G.; Noé, F.; Clementi, C. Machine Learning Of Coarse-grained Molecular Dynamics Force Fields. *ACS Cent. Sci.* **2019**, *5*, 755–767.
- (52) Nüske, F.; Boninsegna, L.; Clementi, C. Coarse-graining Molecular Systems By Spectral Matching. *J. Chem. Phys.* **2019**, *151*, 044116.
- (53) Wang, J.; Chmiela, S.; Müller, K.-R.; Noé, F.; Clementi, C. Ensemble Learning Of Coarse-grained Molecular Dynamics Force Fields With A Kernel Approach. *J. Chem. Phys.* **2020**, *152*, 194106.
- (54) Noé, F.; Tkatchenko, A.; Müller, K.-R.; Clementi, C. Machine Learning For Molecular Simulation. *Annu. Rev. Phys. Chem.* **2020**, *71*, 361–390.
- (55) Schütt, K.; Gastegger, M.; Tkatchenko, A.; Müller, K.-R.; Maurer, R. J. Unifying Machine Learning And Quantum Chemistry With A Deep Neural Network For Molecular Wavefunctions. *Nat. Commun.* **2019**, *10*, 5024.
- (56) Gastegger, M.; McSloy, A.; Luya, M.; Schütt, K.; Maurer, R. A Deep Neural Network For Molecular Wave Functions In Quasi-atomic Minimal Basis Representation. *J. Chem. Phys.* **2020**, *153*, 044123.
- (57) Stöhr, M.; Sandonas, L. M.; Tkatchenko, A. Accurate Many-body Repulsive Potentials For Density-functional Tight-binding From Deep Tensor Neural Networks. *J. Phys. Chem. Lett.* **2020**, *11*, 6835–6843.
- (58) Snyder, J. C.; Rupp, M.; Hansen, K.; Müller, K.-R.; Burke, K. Finding Density Functionals With Machine Learning. *Phys. Rev. Lett.* **2012**, *108*, 253002.
- (59) Brockherde, F.; Vogt, L.; Li, L.; Tuckerman, M. E.; Burke, K.; Müller, K.-R. Bypassing The Kohn-sham Equations With Machine Learning. *Nat. Commun.* **2017**, *8*, 872.
- (60) Bogojeski, M.; Vogt-Maranto, L.; Tuckerman, M. E.; Müller, K.-R.; Burke, K. Quantum Chemical Accuracy From Density Functional Approximations Via Machine Learning. *Nat. Commun.* **2020**, *11*, 5223.
- (61) Hermann, J.; Schätzle, Z.; Noé, F. Deep-neural-network Solution Of The Electronic Schrödinger Equation. *Nat. Chem.* **2020**, *12*, 891–897.
- (62) Carleo, G.; Troyer, M. Solving The Quantum Many-body Problem With Artificial Neural Networks. *Science* **2017**, *355*, 602–606.



- (63) Gebauer, N. W.; Gastegger, M.; Schütt, K. T. Generating Equilibrium Molecules With Deep Neural Networks. *NeurIPS 2018 Workshop on Machine Learning for Molecules and Materials* **2018**.
- (64) Gebauer, N.; Gastegger, M.; Schütt, K. Symmetry-adapted Generation Of 3d Point Sets For The Targeted Discovery Of Molecules. *Adv. Neural. Inf. Process. Syst.* **2019**, 7566–7578.
- (65) Hoffmann, M.; Noé, F. Generating Valid Euclidean Distance Matrices. *arXiv preprint arXiv:1910.03131* **2019**.
- (66) Winter, R.; Montanari, F.; Steffen, A.; Briem, H.; Noé, F.; Clevert, D.-A. Efficient Multi-objective Molecular Optimization In A Continuous Latent Space. *Chem. Sci.* **2019**, 10, 8016–8024.
- (67) Simm, G. N.; Pinsler, R.; Hernández-Lobato, J. M. Reinforcement Learning For Molecular Design Guided By Quantum Mechanics. *arXiv preprint arXiv:2002.07717* **2020**.
- (68) Strieth-Kalthoff, F.; Sandfort, F.; Segler, M. H.; Glorius, F. Machine Learning The Ropes: Principles, Applications And Directions In Synthetic Chemistry. *Chem. Soc. Rev.* **2020**, 49, 6154–6168.
- (69) Chmiela, S.; Sauceda, H. E.; Müller, K.-R.; Tkatchenko, A. Towards Exact Molecular Dynamics Simulations With Machine-learned Force Fields. *Nat. Commun.* **2018**, 9, 3887.
- (70) Sauceda, H. E.; Chmiela, S.; Poltavsky, I.; Müller, K.-R.; Tkatchenko, A. Molecular Force Fields With Gradient-domain Machine Learning: Construction And Application To Dynamics Of Small Molecules With Coupled Cluster Forces. *J. Chem. Phys.* **2019**, 150, 114102.
- (71) Gastegger, M.; Behler, J.; Marquetand, P. Machine Learning Molecular Dynamics For The Simulation Of Infrared Spectra. *Chem. Sci.* **2017**, 8, 6924–6935.
- (72) Christensen, A. S.; Faber, F. A.; von Lilienfeld, O. A. Operators In Quantum Machine Learning: Response Properties In Chemical Space. *J. Chem. Phys.* **2019**, 150, 064105.
- (73) Käser, S.; Unke, O.; Meuwly, M. Reactive Dynamics And Spectroscopy Of Hydrogen Transfer From Neural Network-based Reactive Potential Energy Surfaces. *New J. Phys.* **2020**, 22, 55002.
- (74) Agnihotri, N. Computational Studies Of Charge Transfer In Organic Solar Photovoltaic Cells: A Review. *J. Photochem. Photobiol., C* **2014**, 18, 18–31.
- (75) Payne, M.; Joannopoulos, J.; Allan, D.; Teter, M.; Vanderbilt, D. H. Molecular Dynamics And Ab Initio Total Energy Calculations. *Phys. Rev. Lett.* **1986**, 56, 2656.
- (76) Adcock, S. A.; McCammon, J. A. Molecular Dynamics: Survey Of Methods For Simulating The Activity Of Proteins. *Chem. Rev.* **2006**, 106, 1589–1615.
- (77) Paquet, E.; Viktor, H. L. Molecular Dynamics, Monte Carlo Simulations, And Langevin Dynamics: A Computational Review. *BioMed Res. Int.* **2015**, 2015, 1.
- (78) Schrödinger, E. An Undulatory Theory Of The Mechanics Of Atoms And Molecules. *Phys. Rev.* **1926**, 28, 1049–1070.
- (79) Born, M.; Oppenheimer, R. Zur Quantentheorie der Molekeln. *Ann. Phys.* **1927**, 389, 457–484.
- (80) Meng, X.-Y.; Zhang, H.-X.; Mezei, M.; Cui, M. Molecular Docking: A Powerful Approach For Structure-based Drug Discovery. *Curr. Comput.-Aided Drug Des.* **2011**, 7, 146–157.
- (81) Tuckerman, M. E.; Berne, B. J.; Martyna, G. J.; Klein, M. L. Efficient Molecular Dynamics And Hybrid Monte Carlo Algorithms For Path Integrals. *J. Chem. Phys.* **1993**, 99, 2796–2808.
- (82) Chandler, D.; Wolynes, P. G. Exploiting The Isomorphism Between Quantum Theory And Classical Statistical Mechanics Of Polyatomic Fluids. *J. Chem. Phys.* **1981**, 74, 4078–4095.
- (83) Habershon, S.; Manolopoulos, D. E.; Markland, T. E.; Miller, T. F., III Ring-polymer Molecular Dynamics: Quantum Effects In Chemical Dynamics From Classical Trajectories In An Extended Phase Space. *Annu. Rev. Phys. Chem.* **2013**, 64, 387–413.
- (84) Poltavsky, I.; Tkatchenko, A. Modeling Quantum Nuclei With Perturbed Path Integral Molecular Dynamics. *Chem. Sci.* **2016**, 7, 1368–1372.
- (85) Verlet, L. Computer “Experiments” On Classical Fluids I. Thermodynamical Properties Of Lennard-Jones Molecules. *Phys. Rev.* **1967**, 159, 98.
- (86) Noether, E. Invarianten Beliebiger Differentialausdrücke. *Gött. Nachr., mathematisch-physikalische Klasse* **1918**, 1918, 37–44.
- (87) Shannon, C. E. Communication In The Presence Of Noise. *Proc. IRE* **1949**, 37, 10–21.
- (88) Schaffer, J. What Not To Multiply Without Necessity. *Australas. J. Philos.* **2015**, 93, 644–664.
- (89) Geman, S.; Bienenstock, E.; Doursat, R. Neural Networks And The Bias/Variance Dilemma. *Neural Comput.* **1992**, 4, 1–58.
- (90) Müller, K.-R.; Mika, S.; Rätsch, G.; Tsuda, K.; Schölkopf, B. An Introduction To Kernel-based Learning Algorithms. *IEEE Trans. Neural Netw.* **2001**, 12, 181–201.
- (91) Schölkopf, B.; Smola, A.; Müller, K.-R. Kernel Principal Component Analysis. *International Conference on Artificial Neural Networks*. **1997**, 1327, 583–588.
- (92) Schölkopf, B.; Smola, A.; Müller, K.-R. Nonlinear Component Analysis As A Kernel Eigenvalue Problem. *Neural Comput.* **1998**, 10, 1299–1319.
- (93) Schölkopf, B.; Mika, S.; Burges, C. J.; Knirsch, P.; Müller, K.-R.; Rätsch, G.; Smola, A. J. Input Space Versus Feature Space In Kernel-based Methods. *IEEE Trans. Neural Netw.* **1999**, 10, 1000–1017.
- (94) Bishop, C. M. *Neural Networks for Pattern Recognition*; Oxford University Press, 1995.
- (95) Müller, K.-R.; Smola, A. J.; Rätsch, G.; Schölkopf, B.; Kohlmorgen, J.; Vapnik, V. Predicting Time Series With Support Vector Machines. *International Conference on Artificial Neural Networks*. **1997**, 1327, 999–1004.
- (96) Boser, B. E.; Guyon, I. M.; Vapnik, V. N. A Training Algorithm For Optimal Margin Classifiers. *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*. **1992**, 144–152.
- (97) Theodoridis, S.; Koutroumbas, K., et al. *Pattern Recognition*; Elsevier, 2008.
- (98) Theodoridis, S. *Machine Learning: A Bayesian And Optimization Perspective*, 2nd ed.; Academic Press, 2020.
- (99) Kamath, A.; Vargas-Hernández, R. A.; Krems, R. V.; Carrington, T., Jr; Manzhos, S. Neural Networks vs Gaussian Process Regression For Representing Potential Energy Surfaces: A Comparative Study Of Fit Quality And Vibrational Spectrum Accuracy. *J. Chem. Phys.* **2018**, 148, 241702.
- (100) Wolpert, D. H. The Lack Of A Priori Distinctions Between Learning Algorithms. *Neural Comput.* **1996**, 8, 1341–1390.
- (101) Lee, J.; Bahri, Y.; Novak, R.; Schoenholz, S. S.; Pennington, J.; Sohl-Dickstein, J. Deep Neural Networks As Gaussian Processes. *arXiv preprint arXiv:1711.00165* **2017**.
- (102) Matthews, A. G. d. G.; Rowland, M.; Hron, J.; Turner, R. E.; Ghahramani, Z. Gaussian Process Behaviour In Wide Deep Neural Networks. *arXiv preprint arXiv:1804.11271* **2018**.
- (103) Braun, M. L.; Buhmann, J. M.; Müller, K.-R. On Relevant Dimensions In Kernel Feature Spaces. *J. Mach. Learn. Res.* **2008**, 9, 1875–1908.
- (104) Montavon, G.; Braun, M. L.; Müller, K.-R. Kernel Analysis of Deep Networks. *J. Mach. Learn. Res.* **2011**, 12.
- (105) Chmiela, S.; Tkatchenko, A.; Sauceda, H. E.; Poltavsky, I.; Schütt, K. T.; Müller, K.-R. Machine Learning Of Accurate Energy-conserving Molecular Force Fields. *Sci. Adv.* **2017**, 3, No. e1603015.
- (106) Faber, F. A.; Christensen, A. S.; Huang, B.; von Lilienfeld, O. A. Alchemical And Structural Distribution Based Representation For Universal Quantum Machine Learning. *J. Chem. Phys.* **2018**, 148, 241717.
- (107) Christensen, A. S.; Bratholm, L. A.; Faber, F. A.; Anatole von Lilienfeld, O. FCHL Revisited: Faster And More Accurate Quantum Machine Learning. *J. Chem. Phys.* **2020**, 152, 044107.
- (108) Unke, O. T.; Meuwly, M. PhysNet: A Neural Network For Predicting Energies, Forces, Dipole Moments And Partial Charges. *J. Chem. Theory Comput.* **2019**, 15, 3678–3693.
- (109) Schütt, K.; Kindermans, P.-J.; Sauceda, H. E.; Chmiela, S.; Tkatchenko, A.; Müller, K.-R. SchNet: A Continuous-filter Convolutional



tional Neural Network For Modeling Quantum Interactions. *Adv. Neural. Inf. Process. Syst.* **2017**, 991–1001.

(110) Klicpera, J.; Groß, J.; Günnemann, S. Directional Message Passing For Molecular Graphs. *International Conference on Learning Representations (ICLR)* 2020.

(111) Zhang, Y.; Hu, C.; Jiang, B. Embedded Atom Neural Network Potentials: Efficient And Accurate Machine Learning With A Physically Inspired Representation. *J. Phys. Chem. Lett.* **2019**, 10, 4962–4967.

(112) Zhang, L.; Han, J.; Wang, H.; Car, R.; Weinan, E. Deep Potential Molecular Dynamics: A Scalable Model With The Accuracy Of Quantum Mechanics. *Phys. Rev. Lett.* **2018**, 120, 143001.

(113) Zhang, L.; Han, J.; Wang, H.; Saidi, W.; Car, R.; Weinan, E. End-to-end Symmetry Preserving Inter-atomic Potential Energy Model For Finite And Extended Systems. *Adv. Neural. Inf. Process. Syst.* **2018**, 4436–4446.

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

(115) Lubbers, N.; Smith, J. S.; Barros, K. Hierarchical Modeling Of Molecular Energies Using A Deep Neural Network. *J. Chem. Phys.* **2018**, 148, 241715.

(116) Behler, J. Atom-Centered Symmetry Functions for Constructing High-Dimensional Neural Network Potentials. *J. Chem. Phys.* **2011**, 134, 074106.

(117) Bartók, A. P.; Kondor, R.; Csányi, G. On Representing Chemical Environments. *Phys. Rev. B: Condens. Matter Mater. Phys.* **2013**, 87, 184115.

(118) Schütt, K.; Glawe, H.; Brockherde, F.; Sanna, A.; Müller, K.; Gross, E. How To Represent Crystal Structures For Machine Learning: Towards Fast Prediction Of Electronic Properties. *Phys. Rev. B: Condens. Matter Mater. Phys.* **2014**, 89, 205118.

(119) Faber, F.; Lindmaa, A.; von Lilienfeld, O. A.; Armiento, R. Crystal Structure Representations For Machine Learning Models Of Formation Energies. *Int. J. Quantum Chem.* **2015**, 115, 1094–1101.

(120) Faber, F. A.; Lindmaa, A.; Von Lilienfeld, O. A.; Armiento, R. Machine Learning Energies of 2 Million Elpasolite (ABC<sub>2</sub>D<sub>6</sub>) Crystals. *Phys. Rev. Lett.* **2016**, 117, 135502.

(121) Musil, F.; Grisafi, A.; Bartók, A. P.; Ortner, C.; Csányi, G.; Ceriotti, M. Physics-inspired Structural Representations for Molecules and Materials. *arXiv preprint arXiv:2101.04673* **2021**.

(122) Wahba, G. *Spline Models For Observational Data*; Siam, 1990; Vol. 59.

(123) Schölkopf, B.; Herbrich, R.; Smola, A. J. A Generalized Representer Theorem. *International Conference on Computational Learning Theory.* **2001**, 2111, 416–426.

(124) Argyriou, A.; Micchelli, C. A.; Pontil, M. When Is There A Representer Theorem? Vector Versus Matrix Regularizers. *J. Mach. Learn. Res.* **2009**, 10, 2507–2529.

(125) Berlinet, A.; Thomas-Agnan, C. *Reproducing Kernel Hilbert Spaces in Probability and Statistics*; Springer Science & Business Media: Dordrecht, 2011.

(126) Schölkopf, B.; Smola, A. J. *Learning With Kernels: Support Vector Machines, Regularization, Optimization, And Beyond*; MIT Press, 2002.

(127) Rasmussen, C. E. *Summer School on Machine Learning* **2004**, 3176, 63–71.

(128) Murphy, K. P. *Machine Learning: A Probabilistic Perspective*; MIT Press, 2012.

(129) Smola, A. J.; Schölkopf, B.; Müller, K.-R. The Connection Between Regularization Operators And Support Vector Kernels. *Neural Netw.* **1998**, 11, 637–649.

(130) Micchelli, C. A.; Xu, Y.; Zhang, H. Universal Kernels. *J. Mach. Learn. Res.* **2006**, 7, 2651–2667.

(131) Huang, B.; Von Lilienfeld, O. A. Communication: Understanding Molecular Representations In Machine Learning: The Role Of Uniqueness And Target Similarity. *J. Chem. Phys.* **2016**, 145, 161102.

(132) Hofmann, T.; Schölkopf, B.; Smola, A. J. Kernel Methods In Machine Learning. *Ann. Stat.* **2008**, 36, 1171–1220.

(133) Golub, G. H.; Van Loan, C. F. *Matrix Computations*; JHU Press: Baltimore, 2012; Vol. 3.

(134) Raykar, V. C.; Duraiswami, R. Fast Large Scale Gaussian Process Regression Using Approximate Matrix-vector Products. *Learning Workshop* **2007**.

(135) Williams, C. K.; Seeger, M. Using The Nyström Method To Speed Up Kernel Machines. *Adv. Neural. Inf. Process. Syst.* **2001**, 682–688.

(136) Quiñero-Candela, J.; Rasmussen, C. E. A Unifying View Of Sparse Approximate Gaussian Process Regression. *J. Mach. Learn. Res.* **2005**, 6, 1939–1959.

(137) Snelson, E.; Ghahramani, Z. Sparse Gaussian Processes Using Pseudo-inputs. *Adv. Neural. Inf. Process. Syst.* **2006**, 1257–1264.

(138) Rahimi, A.; Recht, B. Random Features For Large-scale Kernel Machines. *Adv. Neural. Inf. Process. Syst.* **2008**, 1177–1184.

(139) Rudi, A.; Carratino, L.; Rosasco, L. Falcon: An Optimal Large Scale Kernel Method. *Adv. Neural. Inf. Process. Syst.* **2017**, 3888–3898.

(140) Moore, E. H. On The Reciprocal Of The General Algebraic Matrix. *Bull. Am. Math. Soc.* **1920**, 26, 394–395.

(141) Penrose, R. A Generalized Inverse For Matrices. *Math. Proc. Cambridge Philos. Soc.* **1955**, 51, 406–413.

(142) Cutajar, K.; Osborne, M.; Cunningham, J.; Filippone, M. Preconditioning Kernel Matrices. *International Conference on Machine Learning.* **2016**, 2529–2538.

(143) Tikhonov, A. N.; Arsenin, V. I.; John, F. *Solutions of Ill-Posed Problems*; Winston: Washington, DC, 1977; Vol. 14.

(144) McCulloch, W. S.; Pitts, W. A Logical Calculus of the Ideas Immanent in Nervous Activity. *Bull. Math. Biophys.* **1943**, 5, 115–133.

(145) Kohonen, T. An Introduction to Neural Computing. *Neural Netw.* **1988**, 1, 3–16.

(146) Abdi, H. A Neural Network Primer. *J. Biol. Syst.* **1994**, 2, 247–281.

(147) Clark, J. W. *Scientific Applications of Neural Nets*; Springer, 1999 1–96.

(148) Ripley, B. D. *Pattern Recognition and Neural Networks*; Cambridge University Press, 2007.

(149) Haykin, S. S. *Neural Networks and Learning Machines*; Pearson: Upper Saddle River, NJ, 2009; Vol. 3.

(150) LeCun, Y. A.; Bottou, L.; Orr, G. B.; Müller, K.-R. *Neural Networks: Tricks of the Trade*; Springer, 2012; pp 9–48.

(151) Cybenko, G. Approximation By Superposition Of Sigmoidal Functions. *Math. Control Signals Syst.* **1989**, 2, 303–314.

(152) Hornik, K. Approximation Capabilities of Multilayer Feedforward Networks. *Neural Netw.* **1991**, 4, 251–257.

(153) Eldan, R.; Shamir, O. The Power of Depth for Feedforward Neural Networks. *Conference on Learning Theory.* **2016**, 907–940.

(154) Cohen, N.; Sharir, O.; Shashua, A. On The Expressive Power Of Deep Learning: A Tensor Analysis. *Conference On Learning Theory.* **2016**, 698–728.

(155) Telgarsky, M. Benefits Of Depth In Neural Networks. *Conference On Learning Theory.* **2016**, 1517–1539.

(156) Lu, Z.; Pu, H.; Wang, F.; Hu, Z.; Wang, L. The Expressive Power Of Neural Networks: A View From The Width. *Adv. Neural. Inf. Process. Syst.* **2017**, 30, 6231–6239.

(157) Montavon, G.; Orr, G.; Müller, K.-R. *Neural Networks: Tricks Of The Trade*; Springer LNCS 7700, 2012; Vol. 2.

(158) Snoek, J.; Larochelle, H.; Adams, R. P. Practical Bayesian Optimization Of Machine Learning Algorithms. *Adv. Neural. Inf. Process. Syst.* **2012**, 25, 2951–2959.

(159) Hastie, T.; Tibshirani, R.; Friedman, J. *The Elements Of Statistical Learning: Data Mining, Inference, and Prediction*; Springer Science & Business Media, 2009.

(160) Schütt, K. T.; Arbabzadah, F.; Chmiela, S.; Müller, K. R.; Tkatchenko, A. Quantum-Chemical Insights from Deep Tensor Neural Networks. *Nat. Commun.* **2017**, 8, 13890.

- (161) Saucedo, H. E.; Vassilev-Galindo, V.; Chmiela, S.; Müller, K.-R.; Tkatchenko, A. Dynamical Strengthening Of Covalent And Non-covalent Molecular Interactions By Nuclear Quantum Effects At Finite Temperature. *Nat. Commun.* **2021**, *12*, 442.
- (162) Anselmi, F.; Rosasco, L.; Poggio, T. On Invariance And Selectivity In Representation Learning. *Information and Inference: A Journal of the IMA* **2016**, *5*, 134–158.
- (163) Hellman, H. *Einführung In Die Quantenchemie*; Leipzig: F. Deuticke, 1937; Vol. 0.
- (164) Feynman, R. P. Forces In Molecules. *Phys. Rev.* **1939**, *56*, 340.
- (165) Chmiela, S.; Saucedo, H. E.; Poltavsky, I.; Müller, K.-R.; Tkatchenko, A. sGDML: Constructing Accurate And Data Efficient Molecular Force Fields Using Machine Learning. *Comput. Phys. Commun.* **2019**, *240*, 38–45.
- (166) Adamo, C.; Barone, V. Toward Reliable Density Functional Methods Without Adjustable Parameters: The PBE0 Model. *J. Chem. Phys.* **1999**, *110*, 6158–6170.
- (167) Perdew, J. P.; Burke, K.; Ernzerhof, M. Generalized Gradient Approximation Made Simple. *Phys. Rev. Lett.* **1996**, *77*, 3865–3868.
- (168) Tkatchenko, A.; Scheffler, M. Accurate Molecular Van Der Waals Interactions From Ground-state Electron Density And Free-atom Reference Data. *Phys. Rev. Lett.* **2009**, *102*, 073005.
- (169) Montavon, G.; Hansen, K.; Fazli, S.; Rupp, M.; Biegler, F.; Ziehe, A.; Tkatchenko, A.; Lilienfeld, A. V.; Müller, K.-R. Learning Invariant Representations Of Molecules For Atomization Energy Prediction. *Adv. Neural Inf. Process. Syst.* **2012**, 440–448.
- (170) Huo, H.; Rupp, M. Unified Representation Of Molecules And Crystals For Machine Learning. *arXiv preprint arXiv:1704.06439* **2017**.
- (171) Hirn, M.; Mallat, S.; Poilvert, N. Wavelet Scattering Regression Of Quantum Chemical Energies. *Multiscale Model. Simul.* **2017**, *15*, 827–863.
- (172) Eickenberg, M.; Exarchakis, G.; Hirn, M.; Mallat, S. Solid Harmonic Wavelet Scattering: Predicting Quantum Molecular Energy From Invariant Descriptors Of 3d Electronic Densities. *Adv. Neural Inf. Process. Syst.* **2017**, 6540–6549.
- (173) Kriege, N. M.; Giscard, P.-L.; Wilson, R. On Valid Optimal Assignment Kernels And Applications To Graph Classification. *Adv. Neural Inf. Process. Syst.* **2016**, 1623–1631.
- (174) Vert, J. The Optimal Assignment Kernel Is Not Positive Definite. *arXiv preprint arXiv:0801.4061* **2008**.
- (175) Pachauri, D.; Kondor, R.; Singh, V. *Adv. Neural Inf. Process. Syst.* **2013**, 1860–1868.
- (176) Bartók, A. P.; Csányi, G. Gaussian Approximation Potentials: A Brief Tutorial Introduction. *Int. J. Quantum Chem.* **2015**, *115*, 1051–1057.
- (177) De, S.; Bartók, A. P.; Csányi, G.; Ceriotti, M. Comparing Molecules And Solids Across Structural And Alchemical Space. *Phys. Chem. Chem. Phys.* **2016**, *18*, 13754–13769.
- (178) Umeyama, S. An Eigendecomposition Approach To Weighted Graph Matching Problems. *IEEE Trans. Pattern Anal. Mach. Intell.* **1988**, *10*, 695–703.
- (179) 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.
- (180) Bartók, A. P.; De, S.; Poelking, C.; Bernstein, N.; Kermode, J. R.; Csányi, G.; Ceriotti, M. Machine Learning Unifies The Modeling Of Materials And Molecules. *Sci. Adv.* **2017**, *3*, No. e1701816.
- (181) Grisafi, A.; Wilkins, D. M.; Csányi, G.; Ceriotti, M. Symmetry-adapted Machine Learning For Tensorial Properties Of Atomistic Systems. *Phys. Rev. Lett.* **2018**, *120*, 036002.
- (182) Csányi, G.; Willatt, M. J.; Ceriotti, M. *Machine Learning Meets Quantum Physics*; Springer, 2020; pp 99–127.
- (183) 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.
- (184) Brown, D. F.; Gibbs, M. N.; Clary, D. C. Combining *Ab Initio* Computations, Neural Networks, And Diffusion Monte Carlo: An Efficient Method To Treat Weakly Bound Molecules. *J. Chem. Phys.* **1996**, *105*, 7597–7604.
- (185) Tafeit, E.; Estelberger, W.; Horejsi, R.; Moeller, R.; Oettl, K.; Vrecko, K.; Reibnegger, G. Neural Networks As A Tool For Compact Representation Of *ab Initio* Molecular Potential Energy Surfaces. *J. Mol. Graphics* **1996**, *14*, 12–18.
- (186) No, K. T.; Chang, B. H.; Kim, S. Y.; Jhon, M. S.; Scheraga, H. A. Description Of The Potential Energy Surface Of The Water Dimer With An Artificial Neural Network. *Chem. Phys. Lett.* **1997**, *271*, 152–156.
- (187) Prudente, F. V.; Neto, J. S. The Fitting Of Potential Energy Surfaces Using Neural Networks Application To The Study Of The Photodissociation Processes. *Chem. Phys. Lett.* **1998**, *287*, 585–589.
- (188) Manzhos, S.; Carrington, T., Jr. A Random-Sampling High Dimensional Model Representation Neural Network For Building Potential Energy Surfaces. *J. Chem. Phys.* **2006**, *125*, 084109.
- (189) Manzhos, S.; Carrington, T., Jr. Using Redundant Coordinates to Represent Potential Energy Surfaces with Lower-Dimensional Functions. *J. Chem. Phys.* **2007**, *127*, 014103.
- (190) Malshe, M.; Narulkar, R.; Raff, L.; Hagan, M.; Bukkapatnam, S.; Agrawal, P.; Komanduri, R. Development of Generalized Potential-Energy Surfaces using Many-Body Expansions, Neural Networks, and Moiety Energy Approximations. *J. Chem. Phys.* **2009**, *130*, 184102.
- (191) Khorshidi, A.; Peterson, A. A. Amp: A Modular Approach To Machine Learning In Atomistic Simulations. *Comput. Phys. Commun.* **2016**, *207*, 310–324.
- (192) Artrith, N.; Urban, A.; Ceder, G. Efficient And Accurate Machine-learning Interpolation Of Atomic Energies In Compositions With Many Species. *Phys. Rev. B: Condens. Matter Mater. Phys.* **2017**, *96*, 014112.
- (193) Unke, O. T.; Meuwly, M. A Reactive, Scalable, And Transferable Model For Molecular Energies From A Neural Network Approach Based On Local Information. *J. Chem. Phys.* **2018**, *148*, 241708.
- (194) Smith, J. S.; Isayev, O.; Roitberg, A. E. ANI-1: An Extensible Neural Network Potential With DFT Accuracy At Force Field Computational Cost. *Chem. Sci.* **2017**, *8*, 3192–3203.
- (195) Yao, K.; Herr, J. E.; Toth, D. W.; Mckintyre, R.; Parkhill, J. The TensorMol-0.1 Model Chemistry: A Neural Network Augmented With Long-range Physics. *Chem. Sci.* **2018**, *9*, 2261–2269.
- (196) Duvenaud, D. K.; Maclaurin, D.; Iparraguirre, J.; Bombarell, R.; Hirzel, T.; Aspuru-Guzik, A.; Adams, R. P. Convolutional Networks On Graphs For Learning Molecular Fingerprints. *Adv. Neural Inf. Process. Syst.* **2015**, 2224–2232.
- (197) Kearnes, S.; McCloskey, K.; Berndl, M.; Pande, V.; Riley, P. Molecular Graph Convolutions: Moving Beyond Fingerprints. *J. Comput.-Aided Mol. Des.* **2016**, *30*, 595–608.
- (198) Gilmer, J.; Schoenholz, S. S.; Riley, P. F.; Vinyals, O.; Dahl, G. E. Neural Message Passing for Quantum Chemistry. *International Conference on Machine Learning*. **2017**, 1263–1272.
- (199) Scarselli, F.; Gori, M.; Tsoi, A. C.; Hagenbuchner, M.; Monfardini, G. The Graph Neural Network Model. *IEEE Trans. Neural Netw.* **2009**, *20*, 61–80.
- (200) Schütt, K. T.; Gastegger, M.; Tkatchenko, A.; Müller, K.-R. Quantum-chemical Insights From Interpretable Atomistic Neural Networks. In *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*. Springer International Publishing: Switzerland, 2019; pp 311–330.
- (201) Hy, T. S.; Trivedi, S.; Pan, H.; Anderson, B. M.; Kondor, R. Predicting Molecular Properties With Covariant Compositional Networks. *J. Chem. Phys.* **2018**, *148*, 241745.
- (202) Anderson, B.; Hy, T. S.; Kondor, R. Cormorant: Covariant Molecular Neural Networks. *Adv. Neural Inf. Process. Syst.* **2019**, 14537–14546.
- (203) Weiler, M.; Geiger, M.; Welling, M.; Boomsma, W.; Cohen, T. S. 3d Steerable CNNs: Learning Rotationally Equivariant Features In Volumetric Data. *Adv. Neural Inf. Process. Syst.* **2018**, 10381–10392.
- (204) Nair, V.; Hinton, G. E. Rectified Linear Units Improve Restricted Boltzmann Machines. *Int. Conf. Mach. Learn.* **2010**.



- (205) Behler, J. Constructing High-dimensional Neural Network Potentials: A Tutorial Review. *Int. J. Quantum Chem.* **2015**, *115*, 1032–1050.
- (206) Gastegger, M.; Schwiedrzik, L.; Bittermann, M.; Berzsényi, F.; Marquetand, P. wACSF – Weighted Atom-centered Symmetry Functions As Descriptors In Machine Learning Potentials. *J. Chem. Phys.* **2018**, *148*, 241709.
- (207) Kondor, R. I.; Lafferty, J. Diffusion Kernels On Graphs And Other Discrete Structures. *Proceedings of the 19th International Conference on Machine Learning*. **2002**, 315–22.
- (208) Vinyals, O.; Bengio, S.; Kudlur, M. Order Matters: Sequence To Sequence For Sets. *arXiv preprint arXiv:1511.06391* **2015**.
- (209) Grimme, S.; Antony, J.; Ehrlich, S.; Krieg, H. A Consistent And Accurate Ab Initio Parametrization Of Density Functional Dispersion Correction (DFT-D) For The 94 Elements H-Pu. *J. Chem. Phys.* **2010**, *132*, 154104.
- (210) Unke, O.; Muwly, M.  $S_N2$  reactions data set. *Zenodo*, **2019**. DOI: 10.5281/zenodo.2605341
- (211) Monticelli, L.; Tieleman, D. P. *Biomolecular simulations*; Springer, 2013 197–213.
- (212) Friesner, R. A. Ab Initio Quantum Chemistry: Methodology And Applications. *Proc. Natl. Acad. Sci. U. S. A.* **2005**, *102*, 6648–6653.
- (213) Brickel, S.; Das, A. K.; Unke, O. T.; Turan, H. T.; Muwly, M. Reactive Molecular Dynamics For The  $[Cl-CH_3-Br]^-$  Reaction In The Gas Phase And In Solution: A Comparative Study Using Empirical And Neural Network Force Fields. *Electron. Struct.* **2019**, *1*, 024002.
- (214) Saucedo, H. E.; Gastegger, M.; Chmiela, S.; Müller, K.-R.; Tkatchenko, A. Molecular Force Fields With Gradient-domain Machine Learning (GDML): Comparison And Synergies With Classical Force Fields. *J. Chem. Phys.* **2020**, *153*, 124109.
- (215) Szalay, P. G.; Muller, T.; Gidofalvi, G.; Lischka, H.; Shepard, R. Multiconfiguration Self-consistent Field And Multireference Configuration Interaction Methods And Applications. *Chem. Rev.* **2012**, *112*, 108–181.
- (216) Jia, W.; Wang, H.; Chen, M.; Lu, D.; Liu, J.; Lin, L.; Car, R.; Zhang, L., et al. Pushing The Limit Of Molecular Dynamics With Ab Initio Accuracy To 100 Million Atoms With Machine Learning. *arXiv preprint arXiv:2005.00223* **2020**.
- (217) Sanders, H.; Saxe, J. Garbage In, Garbage Out: How Purportedly Great ML Models Can Be Screwed Up By Bad Data. *Proceedings of Blackhat*, 2017.
- (218) Blum, V.; Gehrke, R.; Hanke, F.; Havu, P.; Havu, V.; Ren, X.; Reuter, K.; Scheffler, M. Ab Initio Molecular Simulations With Numeric Atom-centered Orbitals. *Comput. Phys. Commun.* **2009**, *180*, 2175–2196.
- (219) Csányi, G.; Albaret, T.; Payne, M.; De Vita, A. “Learn On The Fly”: A Hybrid Classical And Quantum-mechanical Molecular Dynamics Simulation. *Phys. Rev. Lett.* **2004**, *93*, 175503.
- (220) Seung, H. S.; Opper, M.; Sompolinsky, H. Query By Committee. *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*. **1992**, 287–294.
- (221) Morawietz, T.; Sharma, V.; Behler, J. A Neural Network Potential-energy Surface For The Water Dimer Based On Environment-dependent Atomic Energies And Charges. *J. Chem. Phys.* **2012**, *136*, 064103.
- (222) Smith, J. S.; Nebgen, B.; Lubbers, N.; Isayev, O.; Roitberg, A. E. Less Is More: Sampling Chemical Space With Active Learning. *J. Chem. Phys.* **2018**, *148*, 241733.
- (223) Srivastava, N.; Hinton, G.; Krizhevsky, A.; Sutskever, I.; Salakhutdinov, R. Dropout: A Simple Way To Prevent Neural Networks From Overfitting. *J. Mach. Learn. Res.* **2014**, *15*, 1929–1958.
- (224) Gal, Y.; Ghahramani, Z. Dropout As A Bayesian Approximation: Representing Model Uncertainty In Deep Learning. *International Conference on Machine Learning*. **2016**, 1050–1059.
- (225) Li, Z.; Kermode, J. R.; De Vita, A. Molecular Dynamics With On-the-fly Machine Learning Of Quantum-mechanical Forces. *Phys. Rev. Lett.* **2015**, *114*, 096405.
- (226) Gastegger, M.; Marquetand, P. *Machine Learning Meets Quantum Physics*; Springer, 2020; pp 233–252.
- (227) Shapeev, A.; Gubaev, K.; Tsybalov, E.; Podryabinkin, E. *Machine Learning Meets Quantum Physics*; Springer, 2020; pp 309–329.
- (228) Barducci, A.; Bonomi, M.; Parrinello, M. *Metadynamics*. *Wiley Interdiscip. Rev.: Comput. Mol. Sci.* **2011**, *1*, 826–843.
- (229) Herr, J. E.; Yao, K.; McIntyre, R.; Toth, D. W.; Parkhill, J. Metadynamics For Training Neural Network Model Chemistries: A Competitive Assessment. *J. Chem. Phys.* **2018**, *148*, 241710.
- (230) Sugiyama, M.; Krauledat, M.; Müller, K.-R. Covariate Shift Adaptation By Importance Weighted Cross Validation. *J. Mach. Learn. Res.* **2007**, *8*, 985–1005.
- (231) Lemm, S.; Blankertz, B.; Dickhaus, T.; Müller, K.-R. Introduction To Machine Learning For Brain Imaging. *NeuroImage* **2011**, *56*, 387–399.
- (232) Nesterov, Y. E. A Method For Solving The Convex Programming Problem With Convergence Rate. *Proc. USSR Acad. Sci.* **1983**, 543–547.
- (233) Qian, N. On The Momentum Term In Gradient Descent Learning Algorithms. *Neural Netw.* **1999**, *12*, 145–151.
- (234) Duchi, J.; Hazan, E.; Singer, Y. Adaptive Subgradient Methods For Online Learning And Stochastic Optimization. *J. Mach. Learn. Res.* **2011**, 12.
- (235) Zeiler, M. D. Adadelta: An Adaptive Learning Rate Method. *arXiv preprint arXiv:1212.5701* **2012**.
- (236) Ruders, S. An Overview Of Gradient Descent Optimization Algorithms. *arXiv preprint arXiv:1609.04747* **2016**.
- (237) Kingma, D. P.; Ba, J. Adam: A Method For Stochastic Optimization. *International Conference on Learning Representations*. **2015**, 1–13.
- (238) Huber, P. J. *Breakthroughs in statistics*; Springer, 1992; pp 492–518.
- (239) Barron, J. T. A General And Adaptive Robust Loss Function. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. **2019**, 4331–4339.
- (240) Chmiela, S. *Towards Exact Molecular Dynamics Simulations With Invariant Machine-learned Models*; Technische Universität Berlin: Germany, 2019.
- (241) Christensen, A. S.; von Lilienfeld, O. A. On The Role Of Gradients For Machine Learning Of Molecular Energies And Forces. *Machine Learning: Science and Technology* **2020**, *1*, 045018.
- (242) Meyer, R.; Weichselbaum, M.; Hauser, A. W. Machine Learning Approaches Toward Orbital-free Density Functional Theory: Simultaneous Training On The Kinetic Energy Density Functional And Its Functional Derivative. *J. Chem. Theory Comput.* **2020**, *16*, 5685–5694.
- (243) Bergstra, J.; Bengio, Y. Random Search For Hyper-parameter Optimization. *J. Mach. Learn. Res.* **2012**, *13*, 281–305.
- (244) Prechelt, L. *Neural Networks: Tricks of the trade*; Springer, 1998; pp 55–69.
- (245) Schütt, K.; Kessel, P.; Gastegger, M.; Nicoli, K.; Tkatchenko, A.; Müller, K.-R. SchNetPack: A Deep Learning Toolbox For Atomistic Systems. *J. Chem. Theory Comput.* **2019**, *15*, 448–455.
- (246) Hjorth Larsen, A.; Jørgen Mortensen, J.; Blomqvist, J.; Castelli, I. E.; Christensen, R.; Dulak, M.; Friis, J.; Groves, M. N.; Hammer, B.; Hargus, C.; Hermes, E. D.; Jennings, P. C.; Bjerre Jensen, P.; Kermode, J.; Kitchin, J. R.; Leonhard Kolsbjerg, E.; Kubal, J.; Kaasbjerg, K.; Lysgaard, S.; Bergmann Maronsson, J.; Maxson, T.; Olsen, T.; Pastewka, L.; Peterson, A.; Rostgaard, C.; Schiøtz, J.; Schütt, O.; Strange, M.; Thygesen, K. S.; Vegge, T.; Vilhelmsen, L.; Walter, M.; Zeng, Z.; Jacobsen, K. W.; et al. The Atomic Simulation Environment – A Python Library For Working With Atoms. *J. Phys.: Condens. Matter* **2017**, *29*, 273002.



- (247) Kapil, V.; et al. i-PI 2.0: A Universal Force Engine For Advanced Molecular Simulations. *Comput. Phys. Commun.* **2019**, *236*, 214–223.
- (248) Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; et al. PyTorch: An Imperative Style, High-performance Deep Learning Library. *Adv. Neural. Inf. Process. Syst.* **2019**, 8026–8037.
- (249) Ramakrishnan, R.; Dral, P. O.; Rupp, M.; Von Lilienfeld, O. A. Quantum Chemistry Structures And Properties Of 134 Kilo Molecules. *Sci. Data* **2014**, *1*, 1–7.
- (250) Abadi, M.; Barham, P.; Chen, J.; Chen, Z.; Davis, A.; Dean, J.; Devin, M.; Ghemawat, S.; Irving, G.; Isard, M.; et al. Tensorflow: A System For Large-scale Machine Learning. *12th USENIX symposium on operating systems design and implementation (OSDI 16)* **2016**, 265–283.
- (251) Plimpton, S. *Fast Parallel Algorithms For Short-range Molecular Dynamics*; **1993**. DOI: [10.2172/10176421](https://doi.org/10.2172/10176421)
- (252) Wang, H.; Zhang, L.; Han, J.; Weinan, E. DeePMD-kit: A Deep Learning Package For Many-body Potential Energy Representation And Molecular Dynamics. *Comput. Phys. Commun.* **2018**, *228*, 178–184.
- (253) Himanen, L.; Jäger, M. O. J.; Morooka, E. V.; Federici Canova, F.; Ranawat, Y. S.; Gao, D. Z.; Rinke, P.; Foster, A. S. Dscribe: Library Of Descriptors For Machine Learning In Materials Science. *Comput. Phys. Commun.* **2020**, *247*, 106949.
- (254) Christensen, A.; Faber, F.; Huang, B.; Bratholm, L.; Tkatchenko, A.; Müller, K.-R.; von Lilienfeld, O. QML: A Python Toolkit For Quantum Machine Learning. GitHub, 2017. <https://github.com/qmlcode/qml>.
- (255) Unke, O. T.; Meuwly, M. Toolkit For The Construction Of Reproducing Kernel-based Representations Of Data: Application To Multidimensional Potential Energy Surfaces. *J. Chem. Inf. Model.* **2017**, *57*, 1923–1931.
- (256) Boltzmann, L. *Vorlesungen über Gastheorie. 2. Teil*; Leipzig: J. A. Barth, 1898.
- (257) Botu, V.; Batra, R.; Chapman, J.; Ramprasad, R. Machine Learning Force Fields: Construction, Validation, And Outlook. *J. Phys. Chem. C* **2017**, *121*, 511–522.
- (258) Behler, J. First Principles Neural Network Potentials For Reactive Simulations Of Large Molecular And Condensed Systems. *Angew. Chem., Int. Ed.* **2017**, *56*, 12828–12840.
- (259) Deringer, V. L.; Caro, M. A.; Csányi, G. Machine Learning Interatomic Potentials As Emerging Tools For Materials Science. *Adv. Mater.* **2019**, *31*, 1902765.
- (260) Morawietz, T.; Singraber, A.; Dellago, C.; Behler, J. How Van Der Waals Interactions Determine The Unique Properties Of Water. *Proc. Natl. Acad. Sci. U. S. A.* **2016**, *113*, 8368–8373.
- (261) Calegari Andrade, M. F.; Ko, H.-Y.; Zhang, L.; Car, R.; Selloni, A. Free Energy Of Proton Transfer At The Water–TiO<sub>2</sub> Interface From Ab Initio Deep Potential Molecular Dynamics. *Chem. Sci.* **2020**, *11*, 2335–2341.
- (262) Deringer, V. L.; Csányi, G. Machine Learning Based Interatomic Potential For Amorphous Carbon. *Phys. Rev. B: Condens. Matter Mater. Phys.* **2017**, *95*, 094203.
- (263) Behler, J.; Martonák, R.; Donadio, D.; Parrinello, M. Pressure-induced Phase Transitions In Silicon Studied By Neural Network-based Metadynamics Simulations. *Phys. Status Solidi B* **2008**, *245*, 2618–2629.
- (264) Bartók, A. P.; Kermode, J.; Bernstein, N.; Csányi, G. Machine Learning A General-purpose Interatomic Potential For Silicon. *Phys. Rev. X* **2018**, *8*, 041048.
- (265) Deringer, V. L.; Bernstein, N.; Bartók, A. P.; Cliffe, M. J.; Kerber, R. N.; Marbella, L. E.; Grey, C. P.; Elliott, S. R.; Csányi, G. Realistic Atomistic Structure Of Amorphous Silicon From Machine-learning-driven Molecular Dynamics. *J. Phys. Chem. Lett.* **2018**, *9*, 2879–2885.
- (266) Bonati, L.; Parrinello, M. Silicon Liquid Structure And Crystal Nucleation From Ab Initio Deep Metadynamics. *Phys. Rev. Lett.* **2018**, *121*, 265701.
- (267) Unke, O. T.; Castro-Palacio, J. C.; Bemish, R. J.; Meuwly, M. Collision-induced Rotational Excitation In (N<sub>2</sub><sup>+</sup>, <sup>2</sup>Σ<sub>g</sub><sup>+</sup>, *v* = 0)–Ar: Comparison Of Computations And Experiment. *J. Chem. Phys.* **2016**, *144*, 224307.
- (268) Denis-Alpizar, O.; Unke, O. T.; Bemish, R. J.; Meuwly, M. Quantum And Quasiclassical Trajectory Studies Of Rotational Relaxation In Ar– Collisions. *Phys. Chem. Chem. Phys.* **2017**, *19*, 27945–27951.
- (269) Lu, D.; Li, J.; Guo, H. Comprehensive Investigations Of The Cl+CH<sub>3</sub>OH → HCl+CH<sub>3</sub>O/CH<sub>2</sub>OH Reaction: Validation Of Experiment And Dynamic Insights. *CCS Chem.* **2020**, *2*, 882–894.
- (270) Sweeny, B. C.; Pan, H.; Kassem, A.; Sawyer, J. C.; Ard, S. G.; Shuman, N. S.; Viggiano, A. A.; Brickel, S.; Unke, O. T.; Upadhyay, M.; et al. Thermal Activation Of Methane By MgO+: Temperature Dependent Kinetics, Reactive Molecular Dynamics Simulations And Statistical Modeling. *Phys. Chem. Chem. Phys.* **2020**, *22*, 8913–8923.
- (271) Käser, S.; Unke, O. T.; Meuwly, M. Isomerization And Decomposition Reactions Of Acetaldehyde Relevant To Atmospheric Processes From Dynamics Simulations On Neural Network-based Potential Energy Surfaces. *J. Chem. Phys.* **2020**, *152*, 214304.
- (272) Rivero, U.; Unke, O. T.; Meuwly, M.; Willitsch, S. Reactive Atomistic Simulations Of Diels-Alder Reactions: The Importance Of Molecular Rotations. *J. Chem. Phys.* **2019**, *151*, 104301.
- (273) Liu, Q.; Zhou, X.; Zhou, L.; Zhang, Y.; Luo, X.; Guo, H.; Jiang, B. Constructing High-dimensional Neural Network Potential Energy Surfaces For Gas–surface Scattering And Reactions. *J. Phys. Chem. C* **2018**, *122*, 1761–1769.
- (274) Schütt, K. T.; Sauceda, H. E.; Kindermans, P.-J.; Tkatchenko, A.; Müller, K.-R. SchNet – A Deep Learning Architecture For Molecules And Materials. *J. Chem. Phys.* **2018**, *148*, 241722.
- (275) Hellström, M.; Ceriotti, M.; Behler, J. Nuclear Quantum Effects In Sodium Hydroxide Solutions From Neural Network Molecular Dynamics Simulations. *J. Phys. Chem. B* **2018**, *122*, 10158–10171.
- (276) Chen, W.-K.; Liu, X.-Y.; Fang, W.-H.; Dral, P. O.; Cui, G. Deep Learning For Nonadiabatic Excited-state Dynamics. *J. Phys. Chem. Lett.* **2018**, *9*, 6702–6708.
- (277) Westermayr, J.; Gastegger, M.; Menger, M. F.; Mai, S.; González, L.; Marquetand, P. Machine Learning Enables Long Time Scale Molecular Photodynamics Simulations. *Chem. Sci.* **2019**, *10*, 8100–8107.
- (278) Westermayr, J.; Gastegger, M.; Marquetand, P. Combining SchNet and SHARC: The SchNarc Machine Learning Approach For Excited-state Dynamics. *J. Phys. Chem. Lett.* **2020**, *11*, 3828–3834.
- (279) Raimbault, N.; Grisafi, A.; Ceriotti, M.; Rossi, M. Using Gaussian Process Regression To Simulate The Vibrational Raman Spectra Of Molecular Crystals. *New J. Phys.* **2019**, *21*, 105001.
- (280) Sommers, G. M.; Andrade, M. F. C.; Zhang, L.; Wang, H.; Car, R. Raman Spectrum And Polarizability Of Liquid Water From Deep Neural Networks. *Phys. Chem. Chem. Phys.* **2020**, *22*, 10592–10602.
- (281) Hermann, J.; DiStasio, R. A., Jr; Tkatchenko, A. First-principles Models For Van Der Waals Interactions In Molecules And Materials: Concepts, Theory, And Applications. *Chem. Rev.* **2017**, *117*, 4714–4758.
- (282) Guillot, B. A Reappraisal Of What We Have Learnt During Three Decades Of Computer Simulations On Water. *J. Mol. Liq.* **2002**, *101*, 219–260.
- (283) Møller, C.; Plesset, M. S. Note On An Approximation Treatment For Many-electron Systems. *Phys. Rev.* **1934**, *46*, 618.
- (284) Dunning, T. H., Jr Gaussian Basis Sets For Use In Correlated Molecular Calculations. I. The Atoms Boron Through Neon And Hydrogen. *J. Chem. Phys.* **1989**, *90*, 1007–1023.
- (285) Unke, O. T.; Brickel, S.; Meuwly, M. Sampling Reactive Regions In Phase Space By Following The Minimum Dynamic Path. *J. Chem. Phys.* **2019**, *150*, 074107.
- (286) Manzhos, S.; Carrington, T. Neural Network Potential Energy Surfaces For Small Molecules And Reactions. *Chem. Rev.* **2020**. DOI: [10.1021/acs.chemrev.0c00665](https://doi.org/10.1021/acs.chemrev.0c00665)

- (287) Spura, T.; Elgabarty, H.; Kühne, T. D. "On The Fly" Coupled Cluster Path-integral Molecular Dynamics: Impact Of Nuclear Quantum Effects On The Protonated Water Dimer. *Phys. Chem. Chem. Phys.* **2015**, *17*, 14355–14359.
- (288) Wang, L.; Fried, S. D.; Markland, T. E. Proton Network Flexibility Enables Robustness And Large Electric Fields In The Ketosteroid Isomerase Active Site. *J. Phys. Chem. B* **2017**, *121*, 9807–9815.
- (289) Litman, Y.; Richardson, J. O.; Kumagai, T.; Rossi, M. Elucidating The Nuclear Quantum Dynamics Of Intramolecular Double Hydrogen Transfer In Porphycene. *J. Am. Chem. Soc.* **2019**, *141*, 2526–2534.
- (290) Schran, C.; Briec, F.; Marx, D. Converged Colored Noise Path Integral Molecular Dynamics Study Of The Zundel Cation Down To Ultralow Temperatures At Coupled Cluster Accuracy. *J. Chem. Theory Comput.* **2018**, *14*, 5068–5078.
- (291) Heisenberg, W. *Original Scientific Papers Wissenschaftliche Originalarbeiten*; Springer, 1985; pp 478–504.
- (292) Merchant, H. D.; Srivastava, K. K.; Pandey, H. D. Equations Of State And Thermal Expansion Of Alkali Halides. *CRC Crit. Rev. Solid State Sci.* **1973**, *3*, 451–504.
- (293) Kirchner, V.; Heinke, H.; Hommel, D.; Domagala, J. Z.; Leszczynski, M. Thermal Expansion Of Bulk And Homoepitaxial GaN. *Appl. Phys. Lett.* **2000**, *77*, 1434–1436.
- (294) Jiang, H.; Liu, B.; Huang, Y.; Hwang, K. C. Thermal Expansion Of Single Wall Carbon Nanotubes. *J. Eng. Mater. Technol.* **2004**, *126*, 265–270.
- (295) Kim, D. S.; Hellman, O.; Herriman, J.; Smith, H. L.; Lin, J. Y. Y.; Shulumba, N.; Niedziela, J. L.; Li, C. W.; Abernathy, D. L.; Fultz, B. Nuclear Quantum Effect With Pure Anharmonicity And The Anomalous Thermal Expansion Of Silicon. *Proc. Natl. Acad. Sci. U. S. A.* **2018**, *115*, 1992–1997.
- (296) Hermet, P.; Koza, M. M.; Ritter, C.; Reibel, C.; Viennois, R. Origin Of The Highly Anisotropic Thermal Expansion Of The Semiconducting ZnSb And Relations With Its Thermoelectric Applications. *RSC Adv.* **2015**, *5*, 87118–87131.
- (297) Poltavsky, I.; Zheng, L.; Mortazavi, M.; Tkatchenko, A. Quantum Tunneling Of Thermal Protons Through Pristine Graphene. *J. Chem. Phys.* **2018**, *148*, 204707.
- (298) Markland, T. E.; Ceriotti, M. Nuclear Quantum Effects Enter The Mainstream. *Nat. Rev. Chem.* **2018**, *2*, 0109.
- (299) Freitas, R.; Asta, M.; Bulatov, V. V. Quantum Effects On Dislocation Motion From Ring-polymer Molecular Dynamics. *npj Computational Materials* **2018**, *4*, 55.
- (300) Poltavsky, I.; DiStasio, R. A.; Tkatchenko, A. Perturbed Path Integrals In Imaginary Time: Efficiently Modeling Nuclear Quantum Effects In Molecules And Materials. *J. Chem. Phys.* **2018**, *148*, 102325.
- (301) Zhu, C.; Nakamura, H. The Two-state Linear Curve Crossing Problems Revisited. Iii. Analytical Approximations For Stokes Constant And Scattering Matrix: Nonadiabatic Tunneling Case. *J. Chem. Phys.* **1993**, *98*, 6208–6222.
- (302) Westermayr, J.; Marquetand, P. Machine Learning For Electronically Excited States Of Molecules. *Chem. Rev.* **2020**. DOI: 10.1021/acs.chemrev.0c00749
- (303) Blum, L. C.; Raymond, J.-L. 970 Million Druglike Small Molecules For Virtual Screening In The Chemical Universe Database GDB-13. *J. Am. Chem. Soc.* **2009**, *131*, 8732–8733.
- (304) Ruddigkeit, L.; Van Deursen, R.; Blum, L. C.; Raymond, J.-L. Enumeration Of 166 Billion Organic Small Molecules In The Chemical Universe Database GDB-17. *J. Chem. Inf. Model.* **2012**, *52*, 2864–2875.
- (305) Ghasemi, S. A.; Hofstetter, A.; Saha, S.; Goedecker, S. Interatomic Potentials For Ionic Systems With Density Functional Accuracy Based On Charge Densities Obtained By A Neural Network. *Phys. Rev. B: Condens. Matter Mater. Phys.* **2015**, *92*, 045131.
- (306) Ko, T. W.; Finkler, J. A.; Goedecker, S.; Behler, J. A Fourth-Generation High-Dimensional Neural Network Potential With Accurate Electrostatics Including Non-Local Charge Transfer. *Nat. Commun.* **2021**. DOI: 10.1038/s41467-020-20427-2
- (307) Huang, B.; von Lilienfeld, O. A. Quantum machine learning using atom-in-molecule-based fragments selected on the fly. *Nat. Chem.* **2020**, *12*, 945–951.
- (308) Huang, B.; von Lilienfeld, O. A. Quantum Machine Learning Using Atom-in-molecule-based Fragments Selected On The Fly. *Nat. Chem.* **2020**, *12*, 945–951.
- (309) Dewar, M. J.; Thiel, W. Ground States Of Molecules. 38. The MNDO Method. Approximations And Parameters. *J. Am. Chem. Soc.* **1977**, *99*, 4899–4907.
- (310) Hirshfeld, F. L. Bonded-atom Fragments For Describing Molecular Charge Densities. *Theor. Chim. Acta* **1977**, *44*, 129–138.
- (311) Wilkins, D. M.; Grisafi, A.; Yang, Y.; Lao, K. U.; DiStasio, R. A.; Ceriotti, M. Accurate Molecular Polarizabilities With Coupled Cluster Theory and Machine Learning. *Proc. Natl. Acad. Sci. U. S. A.* **2019**, *116*, 3401–3406.
- (312) Morawietz, T.; Behler, J. A Density-Functional Theory-Based Neural Network Potential for Water Clusters Including van der Waals Corrections. *J. Phys. Chem. A* **2013**, *117*, 7356–7366.
- (313) Uteva, E.; Graham, R. S.; Wilkinson, R. D.; Wheatley, R. J. Interpolation Of Intermolecular Potentials Using Gaussian Processes. *J. Chem. Phys.* **2017**, *147*, 161706.
- (314) Li, H.; Collins, C.; Tanha, M.; Gordon, G. J.; Yaron, D. J. A Density Functional Tight Binding Layer For Deep Learning Of Chemical Hamiltonians. *J. Chem. Theory Comput.* **2018**, *14*, 5764–5776.
- (315) Zubatyuk, T.; Nebgen, B.; Lubbers, N.; Smith, J. S.; Zubatyuk, R.; Zhou, G.; Koh, C.; Barros, K.; Isayev, O.; Tretiak, S. Machine Learned Hückel Theory: Interfacing Physics and Deep Neural Networks. *arXiv preprint arXiv:1909.12963* **2019**.
- (316) Lahey, S.-L. J.; Rowley, C. N. Simulating Protein–Ligand Binding With Neural Network Potentials. *Chem. Sci.* **2020**, *11*, 2362–2368.
- (317) Gastegger, M.; Schütt, K. T.; Müller, K.-R. Machine Learning Of Solvent Effects On Molecular Spectra And Reactions. *arXiv preprint arXiv:2010.14942* **2020**.
- (318) Zhang, P.; Shen, L.; Yang, W. Solvation free energy calculations with quantum mechanics/molecular mechanics and machine learning models. *J. Phys. Chem. B* **2019**, *123*, 901–908.
- (319) Bösel, L.; Thürlmann, M.; Riniker, S. Machine Learning In QM/MM Molecular Dynamics Simulations Of Condensed-Phase Systems. *arXiv preprint arXiv:2010.11610* **2020**.
- (320) Nilsson, N. J. *Principles of Artificial Intelligence*; Springer Science & Business Media, 1982.
- (321) LeCun, Y.; Bengio, Y.; Hinton, G. Deep Learning. *Nature* **2015**, *521*, 436–444.
- (322) Schmidhuber, J. Deep Learning In Neural Networks: An Overview. *Neural Netw.* **2015**, *61*, 85–117.
- (323) Goodfellow, I.; Bengio, Y.; Courville, A. *Deep Learning*; MIT Press: Cambridge, 2016.
- (324) Cortes, C.; Vapnik, V. Support-vector Networks. *Machine learning* **1995**, *20*, 273–297.
- (325) Vapnik, V. *The Nature Of Statistical Learning Theory*; Springer, 1995.
- (326) Lapuschkin, S.; Wäldchen, S.; Binder, A.; Montavon, G.; Samek, W.; Müller, K.-R. Unmasking Clever Hans Predictors And Assessing What Machines Really Learn. *Nat. Commun.* **2019**, *10*, 1096.
- (327) Saucedo, H. E.; Chmiela, S.; Poltavsky, I.; Müller, K.-R.; Tkatchenko, A. *Machine Learning Meets Quantum Physics*; Springer, 2020; pp 277–307.