ANNUAL REVIEWS

Annual Review of Physical Chemistry

Machine Learning for Molecular Simulation

Frank Noé,[1,2,3] Alexandre Tkatchenko,[4] Klaus-Robert Müller,[5,6,7] and Cecilia Clementi[1,3,8]

[1]Department of Mathematics and Computer Science, Freie Universität Berlin, 14195 Berlin, Germany; email: frank.noe@fu-berlin.de

[2]Department of Physics, Freie Universität Berlin, 14195 Berlin, Germany

[3]Department of Chemistry and Center for Theoretical Biological Physics, Rice University, Houston, Texas 77005, USA; email: cecilia@rice.edu

[4]Physics and Materials Science Research Unit, University of Luxembourg, 1511 Luxembourg, Luxembourg; email: alexandre.tkatchenko@uni.lu

[5]Department of Computer Science, Technical University Berlin, 10587 Berlin, Germany; email: klaus-robert.mueller@tu-berlin.de

[6]Max-Planck-Institut für Informatik, 66123 Saarbrücken, Germany

[7]Department of Brain and Cognitive Engineering, Korea University, Seoul 136-713, South Korea

[8]Department of Physics, Rice University, Houston, Texas 77005, USA

Annu. Rev. Phys. Chem. 2020. 71:361–90

First published as a Review in Advance on February 24, 2020

The *Annual Review of Physical Chemistry* is online at physchem.annualreviews.org

https://doi.org/10.1146/annurev-physchem-042018-052331

Keywords

machine learning, neural networks, molecular simulation, quantum mechanics, coarse graining, kinetics

Abstract

Machine learning (ML) is transforming all areas of science. The complex and time-consuming calculations in molecular simulations are particularly suitable for an ML revolution and have already been profoundly affected by the application of existing ML methods. Here we review recent ML methods for molecular simulation, with particular focus on (deep) neural networks for the prediction of quantum-mechanical energies and forces, on coarse-grained molecular dynamics, on the extraction of free energy surfaces and kinetics, and on generative network approaches to sample molecular equilibrium structures and compute thermodynamics. To explain these methods and illustrate open methodological problems, we review some important principles of molecular physics and describe how they can be incorporated into ML structures. Finally, we identify and describe a list of open challenges for the interface between ML and molecular simulation.

1. INTRODUCTION

In 1929 Paul Dirac (1, p. 714) stated,

> The underlying physical laws necessary for the mathematical theory of a large part of physics and the whole of chemistry are thus completely known, and the difficulty is only that the exact application of these laws leads to equations much too complicated to be soluble. It therefore becomes desirable that approximate practical methods of applying quantum mechanics should be developed, which can lead to an explanation of the main features of complex atomic systems without too much computation.

Ninety years later, this quote is still state of the art. However, in the past decade, new tools from the rapidly developing field of machine learning (ML) have started to make a significant impact on the development of approximate methods for complex atomic systems, bypassing the direct solution of "equations much too complicated to be soluble."

ML aims at extracting complex patterns and relationships from large data sets, to predict specific properties of the data. A classical application of ML is to the problem of image classification in which descriptive labels need to be associated with images that are presented in terms of sets of pixels. The machine is trained on a large number of examples and learns how to classify new images. The underlying idea is that there exists a complex relationship between the input (the pixels) and the output (the labels) that is unknown in its explicit form but can be inferred by a suitable algorithm. Clearly, such an operating principle can be very useful in the description of atomic and molecular systems as well. We know that atomistic configurations dictate the chemical properties, and the machine can learn to associate the latter with the former without solving first-principle equations, if presented with enough examples.

Although different ML tools are available and have been applied to molecular simulation [e.g., kernel methods (2)], here we mostly focus on the use of neural networks, now often synonymous with the term deep learning. We assume the reader has basic knowledge of ML, and we refer to the literature for an introduction to statistical learning theory (3, 4) and deep learning (5, 6).

One of the first applications of ML in chemistry has been to extract classical potential energy surfaces from quantum mechanical (QM) calculations, to efficiently perform molecular dynamics (MD) simulations that can incorporate quantum effects. The seminal work of Behler & Parrinello (7) in this direction has opened the way to a now rapidly advancing area of research (8–17). In addition to atomistic force fields, effective molecular models at resolution coarser than atomistic can be designed by ML (18–21). Analysis and simulation of MD trajectories have also been affected by ML—for instance, for the definition of optimal reaction coordinates (22–27), the estimation of free energy surfaces (25, 28–30), the construction of Markov state models (24, 26, 31) and dynamic graphical models (32) of molecular kinetics, and the enhancement of MD sampling by learning bias potentials (33–37) or selecting starting configurations by active learning (38–40). Finally, ML can be used to generate samples from the equilibrium distribution of a molecular system without performing MD altogether, as proposed in the recently introduced Boltzmann generators (41). A selection of these topics are reviewed and discussed in this article.

All these different aspects of molecular simulation have evolved independently so far. For instance, ML-generated force fields have mostly been developed and applied on small molecules and ordered solids, while the analysis of MD trajectories is mostly relevant for the simulation of large, flexible molecules like proteins. In order to really revolutionize the field, these tools and methods need to evolve, become more scalable and transferable, and converge into a complete pipeline for the simulation and analysis of molecular systems. There are still some significant challenges toward this goal, as we discuss here, but considering the rapid progress of the past few years, we envision that in the near future ML will have transformed the way molecular systems are studied in silico.

This review focuses on physics-based ML approaches for molecular simulation. ML is also having a big impact in other areas of chemistry without involving a physics-based model—for example, by enabling direct attempts to predict physicochemical or pharmaceutical properties (42–45) or to design materials and molecules with certain desirable properties by using generative learning (46–49). We refer the interested reader to other recent reviews on the subject (50, 51).

The review is organized as follows. We start by describing the most important ML problems and principles for molecular simulation (Section 2). A crucial aspect of the application of ML in molecular simulations is to incorporate physical constraints, and we discuss this for the most commonly used physical symmetries and invariances for molecular systems (Section 3). We then provide examples of specific ML methods and applications for molecular simulation tasks, focusing on deep learning and the neural network architectures involved (Section 4). We conclude by outlining open problems and pointing out possible approaches to their solution (Section 5).

2. MACHINE LEARNING PROBLEMS FOR MOLECULAR SIMULATION

In this section we discuss how several of the open challenges in the simulation of molecular systems can be formulated as ML problems and describe the recent efforts to address them.

2.1. Potential Energy Surfaces

MD and Markov chain Monte Carlo simulations employing classical force fields within the Born–Oppenheimer approximation constitute the cornerstone of contemporary atomistic modeling in chemistry, biology, and materials science. These methods perform importance sampling; that is, in the long run, they sample states $\mathbf{x}$ from the molecular system's equilibrium distribution, which has the general form

$$\mu(\mathbf{x}) \propto e^{-u(\mathbf{x})}. \quad 1.$$

The reduced potential $u(\mathbf{x})$ contains terms that depend on the thermodynamic constraints (e.g., fixed temperature, pressure). In the canonical ensemble (fixed number of particles, volume, and temperature), $u(\mathbf{x}) = U(\mathbf{x})/k_B T$, where $k_B T$ is the thermal energy at temperature T.

However, the predictive power of these simulations is only as good as the underlying potential energy surface (PES). Hence, predictive simulations of properties and functions of molecular systems require an accurate description of the global PES, $U(\mathbf{x})$, where $\mathbf{x}$ indicates the nuclear Cartesian coordinates. All many-body interactions between electrons are encoded in the $U(\mathbf{x})$ function. Although $U(\mathbf{x})$ could be obtained on the fly using explicit ab initio calculations, more efficient approaches that can access long timescales are required to understand relevant phenomena in large molecular systems. A plethora of classical mechanistic approximations to $U(\mathbf{x})$ exist, in which the parameters are typically fitted to a small set of ab initio calculations or experimental data (52–55). Unfortunately, these classical approximations often suffer from a lack of transferability and can yield accurate results only close to the conditions (geometries) they have been fitted to.

Alternatively, sophisticated ML approaches that can accurately reproduce the global PES for elemental materials (7, 9, 14, 56–58) and small molecules (10, 11, 58–62) have recently been developed (see **Figure 1**). Such methods learn a model of the PES, $\hat{U}(\mathbf{x}, \theta)$, where the parameters θ are optimized by energy matching and/or by force matching. In energy matching, an ML model is trained to minimize the loss function

$$L_{\text{ene}} = \sum_i [\hat{U}(\mathbf{x}_i, \theta) - U_i]^2, \quad 2.$$

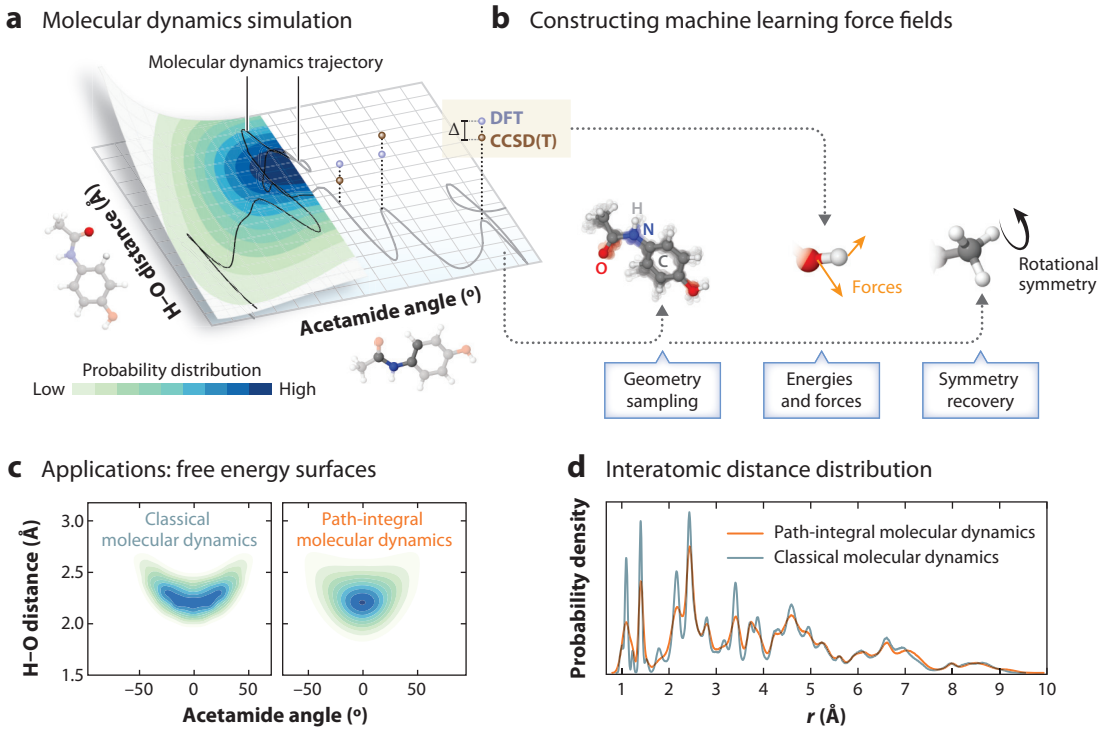

Figure 1

Constructing force field models with machine learning. (*a*) Reference geometries are sampled from a molecular dynamics trajectory that is sufficiently long to ensure optimal coverage of the configuration space. (*b*) For a small subset of geometries, energy and force labels are then computed at a high level of theory to form the training, validation, and test data sets. A globally consistent atom–atom assignment across the whole training set enables the identification and reconstructive exploitation of relevant spatial and temporal physical symmetries of the molecular dynamics. (*c*) The resulting machine learning model of the potential energy surface is finally used to speed up sampling-intensive path-integral molecular dynamics simulations at the accuracy of the reference electronic structure method (10). (*d*) An interatomic distance distribution, sampled with classical and path-integral molecular dynamics, is shown. Abbreviations: CCSD(T), coupled-cluster single double (triple); DFT, density functional theory.

where U_i are energy values obtained by QM calculations at specific configurations (see **Figure 1**). For force matching, we compute the QM forces at specified configurations

$$\mathbf{f}(\mathbf{x}) = -\nabla U(\mathbf{x}) \qquad 3.$$

and minimize the loss function:

$$L_{\text{force}} = \sum_i \left\| \nabla \hat{U}(\mathbf{x}_i, \theta) + \mathbf{f}_i \right\|^2 . \qquad 4.$$

The existing ML PES models are based either on nonlinear kernel learning (9, 10, 57, 61) or on (deep) neural networks (14, 56, 62). Specific neural network architectures are discussed in Sections 4.1 and 4.2. Both approaches have advantages and limitations. The advantage of the kernel methods is that their convex nature yields a unique solution, whose behavior can be controlled outside of the training data. Neural networks are nonconvex ML models and are often harder to interpret and generalize outside of the training manifold.

2.2. Free Energy Surfaces

Given the Cartesian coordinates of a molecule with N atoms, $\mathbf{x} \in \mathbb{R}^{3N}$, we define the collective coordinates by the encoding

$$\mathbf{y} = E(\mathbf{x}), \tag{5.}$$

where $\mathbf{y} \in \mathbb{R}^m$ and m is a small number. In ML terms, the coordinates $\mathbf{y}$ define a latent space. In molecular simulations, the collective coordinates are often defined to describe the slowest processes of the system (63). In general, the mapping (or encoding) of E can be highly nonlinear. If the energy function $U(\mathbf{x})$ associated with the atomistic representation is known, an important quantity to compute—for instance, to connect with experimental measurements—is the free energy of the system as a function of the collective variables. The free energy is defined as

$$F(\mathbf{y}) = -\log \mu_Y(\mathbf{y}) + \text{const}, \tag{6.}$$

where $\mu_Y(\mathbf{y})$ is the marginal distribution on $\mathbf{y}$ of the equilibrium distribution $\mu(\mathbf{x})$ given by Equation 1:

$$\mu_Y(\mathbf{y}) = \int_{\mathbf{x}|E(\mathbf{x})=\mathbf{y}} \mu(\mathbf{x})\mathrm{d}\mathbf{x}. \tag{7.}$$

The integral in Equation 7 is in practice impossible to compute analytically for high-dimensional systems, and several methods have been developed for its numerical estimation by enhancing the sampling of the equilibrium distribution in MD simulations of the system.

The definition of the free energy can also be formulated as a learning problem: The aim is to optimize the parameters θ of a free energy function model $\hat{F}(\mathbf{y}, \theta)$ such that Equations 6 and 7 are satisfied to a good approximation. Fulfilling these equations is usually referred to as enforcing thermodynamic consistency.

Using methods that estimate the free energy F^λ (or its gradient ∇F^λ) at a given set of points in the collective variables space $\mathbf{y}^\lambda$ ($\lambda = 1, \ldots, M$), one can use the free energy loss,

$$L_{\text{ene}} = \sum_{\lambda=1}^{M} \|F^\lambda - \hat{F}(\mathbf{y}^\lambda, \theta)\|^2,$$

or the free energy gradient loss,

$$L_{\text{grad}} = \sum_{\lambda=1}^{M} \|\nabla F^\lambda - \nabla \hat{F}(\mathbf{y}^\lambda, \theta)\|^2,$$

with these free energy estimates to reconstruct the entire surface $\hat{F}(\mathbf{y},\theta)$. Both kernel regression (28) and neural networks (30) have been used for this purpose. ML has also been used in combination with enhanced sampling methods to learn the free energy surface on the fly (25, 29, 64–69).

An alternative way to learn Equation 6 is by using force matching (70, 71). It has been demonstrated (72) that, given the forces $\nabla_{\mathbf{x}} U$ of the atomistic system collected along an MD trajectory, $\mathbf{x}_t$, $t = 1, \ldots, T$, the gradient of a free energy model $\hat{F}(\mathbf{y},\theta)$ that best satisfies Equation 6 also minimizes the force-matching loss,

$$L_{\text{force}} = \sum_t \left\| \nabla_{\mathbf{y}} \hat{F}[E(\mathbf{x}_t), \theta] + f^{\mathbf{y}}_{\text{lmf}}(\mathbf{x}_t) \right\|^2, \tag{8.}$$

where the term $f^{\mathbf{y}}_{\text{lmf}}$ is the local mean force,

$$f^{\mathbf{y}}_{\text{lmf}}(\mathbf{x}) = \nabla_{\mathbf{x}} U \cdot G^{\mathbf{y}} + \nabla_{\mathbf{x}} \cdot G^{\mathbf{y}}$$
$$G^{\mathbf{y}} = \nabla_{\mathbf{x}} E [(\nabla_{\mathbf{x}} E)^T \nabla_{\mathbf{x}} E]^{-1},$$

that is, the projection of the atomistic force $\nabla_{\mathbf{x}} U$ on the collective variable space through the mapping E.

In practice, the estimator in Equation 8 is very noisy: Because of the dimensionality reduction from $\mathbf{x}$ to $\mathbf{y}$, multiple realizations of the projected force $f^{\mathbf{y}}_{\text{lmf}}$ can be associated with the same value of the collective coordinates $\mathbf{y}$, and the minimum of the loss function in Equation 8 cannot go to zero. It can be shown by invoking statistical estimator theory that this loss can be broken down into bias, variance, and noise terms (20).

2.3. Coarse Graining

The use of coarse-grained models of complex molecular systems, such as proteins, presents an attractive alternative to the use of atomistic models that are very expensive to simulate (71, 73).

The design of a coarse-grained model for a system with N atoms into a reduced representation with n effective beads starts with the definition of a mapping similar to Equation 5, where now $\mathbf{y} \in \mathbb{R}^{3n}$ are the coordinates of the coarse-grained beads. In this case, the mapping is usually a linear function, as in general the beads are defined as a subset or a linear combination of sets of atoms in the original system, in a way that allows one to keep some information about the geometry of the molecule. For instance, in protein systems, a coarse-grained mapping can be defined by replacing all the atoms in a residue with a bead centered on the corresponding C_α atom. There is at present no general theory to define the optimal coarse-graining mapping for a specific system. A few methods have been proposed in this direction by optimizing the mapping E to preserve certain properties of the original system. Examples include the definition of a system's dynamical assembly units (74) or the use of an autoencoder to minimize the reconstruction error (75).

Once the coarse-graining mapping is given, several strategies exist to define the model energy function, either to match experimental observables (top-down) or to reproduce specific properties of the atomistic system (bottom-up) (71). If one wants to define a coarse-grained model that is thermodynamically consistent with the original model (Equations 6 and 7), then the definition of the energy function associated with a coarse-grained molecular model can be seen as a special case of the free energy learning discussed in the previous section (70). The effective energy of the coarse-grained model is the free energy defined by Equation 6, where $\mathbf{y}$ are now the coarse variables. Therefore, once the coarse-graining map is defined, the definition of the effective potential can also be seen as a learning problem. In particular, the force-matching loss function given by Equation 8 can be used to train the effective energy of the model from the atomistic forces. For a general linear mapping $\mathbf{y} = \mathbf{E}\mathbf{x}$ where matrix $\mathbf{E} \in \mathbb{R}^{3n \times 3N}$ clusters N atomic coordinates into n bead coordinates, the expression for the loss, Equation 8, reduces to

$$L_{\text{force}} = \sum_t \| \nabla_{\mathbf{y}} \hat{F}(\mathbf{E}\mathbf{x}_t, \theta) + f^{\mathbf{y}}(\mathbf{x}_t) \|^2, \qquad 9.$$

where the instantaneous coarse-grained force is given by $f^{\mathbf{y}}(\mathbf{x}) = (\mathbf{E}\mathbf{E}^\top)^{-1}\mathbf{E}\mathbf{f}(\mathbf{x})$ of the atomistic forces $\mathbf{f}(\mathbf{x})$ (Equation 3). This loss function has been used to design coarse-grained force fields for different systems with both kernel methods (18) and deep neural networks (19, 20).

2.4. Kinetics

Kinetics are the slow part of the dynamics. Due to the stochastic components in the MD integrator, for any trajectory emerging from a configuration $\mathbf{x}_t$ at time t, there is a probability distribution of finding the molecule in configuration $\mathbf{x}_{t+\tau}$ at a later time $t+\tau$:

$$\mathbf{x}_{t+\tau} \sim p_\tau(\mathbf{x}_{t+\tau} \mid \mathbf{x}_t). \tag{10.}$$

We can express the transition density (Equation 10) as the action of the Markov propagator in continuous space and by its spectral decomposition (76, 77):

$$p(\mathbf{x}_{t+\tau}) = \int p(\mathbf{x}_{t+\tau} \mid \mathbf{x}_t; \tau) p(\mathbf{x}_t)\, \mathrm{d}\mathbf{x}_t \approx \sum_{k=1}^{n} \sigma_k^* \langle p(\mathbf{x}_t) \mid \phi(\mathbf{x}_t) \rangle \psi(\mathbf{x}_{t+\tau}). \tag{11.}$$

The spectral decomposition can be read as follows: The evolution of the probability density $p(\mathbf{x})$ can be approximated as the superposition of functions ψ. A second set of functions, ϕ, is required to compute the amplitudes of these functions.

In general, Equation 11 is a singular value decomposition with left and right singular functions ϕ_k, ψ_k and true singular values σ_k^* (77). The approximation is then a low-rank decomposition in which the small singular values are discarded. For the special case that dynamics are in thermal equilibrium, detailed balance (Equation 21) holds, and Equation 11 is an eigenvalue decomposition with the choices

$$\begin{aligned} \sigma_k^* &= \lambda_k^*(\tau) = e^{-\tau/t_i^*}, \\ \phi_k(\mathbf{x}) &= \psi_k(\mathbf{x}) \mu(\mathbf{x}). \end{aligned} \tag{12.}$$

Hence, Equation 11 simplifies: We need only one set of functions, the eigenfunctions ψ_k. The true eigenvalues λ_k^* are real valued and decay exponentially in time τ with characteristic relaxation times t_i^* that are directly linked to kinetic experimental observables (78, 79). The approximation in Equation 11 is due to truncating all terms with relaxation times shorter than t_n^*.

A quite established approach is to learn molecular kinetics (Equation 11) from a given trajectory data set. To obtain a low-dimensional model of the molecular kinetics that is easy to interpret and analyze, this usually involves two steps: (*a*) finding a low-dimensional latent space representation of the collective variables, $\mathbf{y} = E(\mathbf{x})$, using the encoder E; and (*b*) learning a dynamical propagator $\mathbf{P}$ in that space:

$$\begin{array}{ccc} \mathbf{x}_t & \xrightarrow{E} & \mathbf{y}_t \\ \mathrm{MD} \downarrow & & \downarrow \mathbf{P} \\ \mathbf{x}_{t+\tau} & \xrightarrow{E} & \mathbf{y}_{t+\tau}. \end{array} \tag{13.}$$

A common approach in MD, but also in other fields such as dynamical systems and fluid mechanics, is to seek a pair $(E, \mathbf{P})$ such that $\mathbf{P}$ is a small matrix that propagates state vectors in a Markovian (memory-less) fashion (77, 78, 80–88). This is motivated by the spectral decomposition of dynamics (Equation 11): If τ is large enough to filter fast processes, a few functions are sufficient to describe the kinetics of the system, and if E maps to the space spanned by these functions, $\mathbf{P}$ can be a linear, Markovian model.

If no specific constraints are imposed on $\mathbf{P}$, then the minimum regression error of $\mathbf{y}_{t+\tau} = \mathbf{P}\mathbf{y}_t$, the variational approach of Markov processes (VAMP) (77, 89), and maximum likelihood will all

lead to the same estimator (90),

$$\mathbf{P} = \mathbf{C}_{00}^{-1}\mathbf{C}_{0\tau}, \tag{14.}$$

using the latent space covariance matrices

$$\mathbf{C}_{00} = \frac{1}{T}\sum_{t=1}^{T-\tau}\mathbf{y}_t\mathbf{y}_t^\top, \mathbf{C}_{0\tau} = \frac{1}{T}\sum_{t=1}^{T-\tau}\mathbf{y}_t\mathbf{y}_{t+\tau}^\top, \mathbf{C}_{\tau\tau} = \frac{1}{T}\sum_{t=1}^{T-\tau}\mathbf{y}_{t+\tau}\mathbf{y}_{t+\tau}^\top.$$

If E performs a one-hot encoding that indicates which state the system is in, then the pair $(E, \mathbf{P})$ is called a Markov state model (78, 80–84), and $\mathbf{P}$ is a matrix of conditional probabilities to be in state j at time $t + \tau$ given that the system was in state i at time t.

Learning the embedding E is more difficult than learning $\mathbf{P}$, as optimizing E by maximum likelihood or minimal regression error in latent space $\mathbf{y}$ leads to a collapse of E to trivial, uninteresting solutions (90). This problem can be avoided by following a variational approach to approximating the leading terms of the spectral decomposition (Equation 10) (77, 89). The variational approach for conformation dynamics (89) states that for dynamics obeying detailed balance (21), the eigenvalues λ_k of a propagator matrix $\mathbf{P}$ via any encoding $\mathbf{y} = E(\mathbf{x})$ are, in the statistical limit, lower bounds of the true λ_k^*. The VAMP variational principle is more general, as it does not require detailed balance (see Equation 21) and applies to the singular values σ_k:

$$\lambda_k \leq \lambda_k^* \quad \text{(with detailed balance)},$$

$$\sigma_k \leq \sigma_k^* \quad \text{(no detailed balance)}.$$

As VAMP is the more general principle, we can use it to define the loss function for estimating molecular kinetics models:

$$\mathcal{L}_{\mathrm{VAMP}-2}(\{\mathbf{y}_t^0, \mathbf{y}_t^\tau\}) = -\left\| (\mathbf{C}_{00})^{-\frac{1}{2}}\, \mathbf{C}_{0\tau}\, (\mathbf{C}_{\tau\tau})^{-\frac{1}{2}} \right\|_F^2. \tag{15.}$$

If dynamics obey detailed balance, we can use $\mathbf{C}_{\tau\tau} = \mathbf{C}_{00}$ and plug in a symmetric estimate for $\mathbf{C}_{0\tau}$ (26, 90).

2.5. Sampling and Thermodynamics

MD time steps are on the order of one femtosecond (10^{-15} s), while configuration changes governing molecular function are often rare events that may take 10^{-3} to 10^3 s. Even when the potential and the forces generated by single-protein folding and unfolding are evaluated quickly, simulating these processes by direct MD simulation may take years to centuries on a supercomputer. To avoid this sampling problem, ML methods can be employed to learn generation of equilibrium samples from $\mu(\mathbf{x})$ more efficiently, or even to generate statistically independent samples in one shot.

Learning to sample probability distributions is the realm of generative learning (91–94). In the past few years, directed generative networks, such as variational autoencoders (92), generative adversarial networks (93), and flows (94, 95), have received a particular surge of interest. Such networks are trained to transform samples from an easy-to-sample probability distribution, such as Gaussian noise, into realistic samples of the objects of interest. These methods have been used to draw photorealistic images (96, 97), generate speech or music audio (98), and generate chemical structures to design molecules or materials with certain properties (46–49).

If the aim is to learn a probability distribution from sampling data (density estimation) or to learn to efficiently sample from a given probability distribution [Boltzmann generation (41)], one typically faces the challenge of matching the probability distribution of the trained model with a reference.

Matching probability distributions can be achieved by minimizing probability distances. The most commonly used probability distance is the Kullback–Leibler (KL) divergence, also called relative entropy between two distributions q and p:

$$\mathrm{KL}(q \parallel p) = \int q(\mathbf{x}) \left[\log q(\mathbf{x}) - \log p(\mathbf{x})\right] \mathrm{d}\mathbf{x}. \quad 16.$$

In Boltzmann generation (41), we aim to efficiently sample the equilibrium distribution $\mu(\mathbf{x})$ of a many-body system defined by its energy function $u(\mathbf{x})$ (Equation 1). We can learn the parameters $\boldsymbol{\theta}$ of a neural network that generates samples from the distribution $p_X(\mathbf{x}; \boldsymbol{\theta})$ by making its generated distribution similar to the target equilibrium distribution. Choosing $q \equiv p_X(\mathbf{x})$ and $p \equiv \mu(\mathbf{x})$ in Equation 16, we can minimize the $\mathrm{KL}_{\boldsymbol{\theta}}\left[q_X \parallel \mu_X\right]$ with the energy loss (EL):

$$\mathrm{EL} = \mathbb{E}_{\mathbf{x}\sim p_X(\mathbf{x};\boldsymbol{\theta})}\left[\log p_X(\mathbf{x}; \boldsymbol{\theta}) + u(\mathbf{x})\right]. \quad 17.$$

As the network samples from an energy surface $u(\mathbf{x}; \boldsymbol{\theta}) = -\log p_X(\mathbf{x}; \boldsymbol{\theta}) + \mathrm{const}$ (Equation 1), this loss is performing energy matching, similar to what was discussed in Section 2.1. In order to evaluate the loss (Equation 17), we need not only to be able to generate samples $\mathbf{x} \sim p_X(\mathbf{x}; \boldsymbol{\theta})$ from the network but also to be able to compute $p_X(\mathbf{x}; \boldsymbol{\theta})$ for every sample $\mathbf{x}$ (see Section 4.5 for an example).

The reverse case is density estimation. Suppose we have simulation data $\mathbf{x}$, and we want to train the probability distribution $p_X(\mathbf{x}; \boldsymbol{\theta})$ to resemble the data distribution. We choose $q(\mathbf{x})$ as the data distribution and $p \equiv p_X(\mathbf{x}; \boldsymbol{\theta})$ in Equation 16 and exploit that $\mathbb{E}_{\mathbf{x}\sim\mathrm{data}}\left[\log q(\mathbf{x})\right]$ is a constant that does not depend on the parameters $\boldsymbol{\theta}$. Then we can minimize the loss:

$$\mathrm{NL} = -\mathbb{E}_{\mathbf{x}\sim\mathrm{data}}[\log p_X(\mathbf{x}; \boldsymbol{\theta})]. \quad 18.$$

This loss is the negative likelihood (NL) that the model generates the observed sample; hence, minimizing it corresponds to a maximum likelihood approach. Likelihood maximization is abundantly used in ML; in this review we discuss it in Section 4.5.

3. INCORPORATING PHYSICS INTO MACHINE LEARNING

3.1. Why Incorporate Physics?

Compared to working with classical ML problems such as image classification, we have a decisive advantage when working with molecular problems: We know a lot of physical principles that restrict the possible predictions of our machine to the physically meaningful ones.

Let us start with a simple example to illustrate this. We are interested in predicting the potential energy and the atom-wise forces of the diatomic molecule O_2 with positions $\mathbf{x}_1, \mathbf{x}_2$ in a vacuum (without external forces). Independent of the details of our learning algorithm, physics tells us that a few general rules must hold:

1. The energy is invariant when translating or rotating the molecule. We can thus arbitrarily choose the positions $\mathbf{x}_1 = (0, 0, 0)$; $\mathbf{x}_2 = (d, 0, 0)$, and the energy becomes a function of the interatomic distance only: $U(\mathbf{x}) \to U(d)$.

2. Energy is conserved. The energy U and the force of the molecule are related by $\mathbf{F}(\mathbf{x}) = -\nabla U(\mathbf{x})$. Now we can compute the components of the force as $\mathbf{f}_1 = (\frac{\partial U(d)}{\partial d}, 0, 0)$; $\mathbf{f}_2 = (-\frac{\partial U(d)}{\partial d}, 0, 0)$.
3. Identical particles are indistinguishable. The energy is unchanged if we exchange the labels 1 and 2 of the identical oxygen atoms.

In ML, there are two principal approaches when dealing with such invariances or symmetries: (*a*) data augmentation and (*b*) building the invariances into the ML model.

3.2. Data Augmentation

Data augmentation means we learn invariances by heart by artificially generating more training data and applying the known invariances to it. For example, given a training data point for positions and energy/force labels, $(\mathbf{x}; U, \mathbf{f})$, we can augment this data point with translation invariance of energy and force by adding more training data $(\mathbf{x} + \Delta\mathbf{x}; U, \mathbf{f})$ with random displacements $\Delta\mathbf{x}$. Data augmentation makes the ML model more robust and helps us predict the invariances approximately. It is an important ML tool, as it is easy to do, although for certain invariances it is conceptually difficult or computationally expensive to hardwire those invariances into the ML model.

However, data augmentation is statistically inefficient, as additional training data are needed, and inaccurate, because a network that does not have translation invariance hardwired into it will never predict that the energy is exactly constant when translating the molecule. This inaccuracy may lead to unphysical and potentially catastrophic predictions when such an energy model is inserted into an MD integrator.

3.3. Building Physical Constraints into the Machine Learning Model

The more accurate, statistically efficient, and elegant approach to incorporating physical constraints is to directly build them into the ML model. Doing so involves accounting for two related aspects that are both essential to making the learning problem efficient. First are equivariances: The ML model should have the same invariances and symmetries as the modeled physics problem. Second is parameter sharing: Whenever there is an invariance or symmetry, this should be reflected by sharing model parameters in different parts of the network.

3.4. Invariance and Equivariance

If we can hardwire the physical symmetries into the ML structure, we can reduce the dimensionality of the problem. In the example of the O_2 molecule above, the energy and force are one- and six-dimensional functions defined for each point $\mathbf{x}$ in six-dimensional configuration space. However, when we use the invariances described above, we have to learn only a one-dimensional energy function over a one-dimensional space, and we can compute the full force field from it. The learning problem has become much simpler because we now learn only in the manifold of physically meaningful solutions (**Figure 2*a***).

An important concept related to invariance is equivariance. A function is equivariant if it transforms the same way as its argument. For example, the force is defined by the negative gradient of the energy, $-\nabla U(\mathbf{x})$. If we rotate the molecule by applying the rotation $\mathbf{R}\cdot$, the energy is invariant,

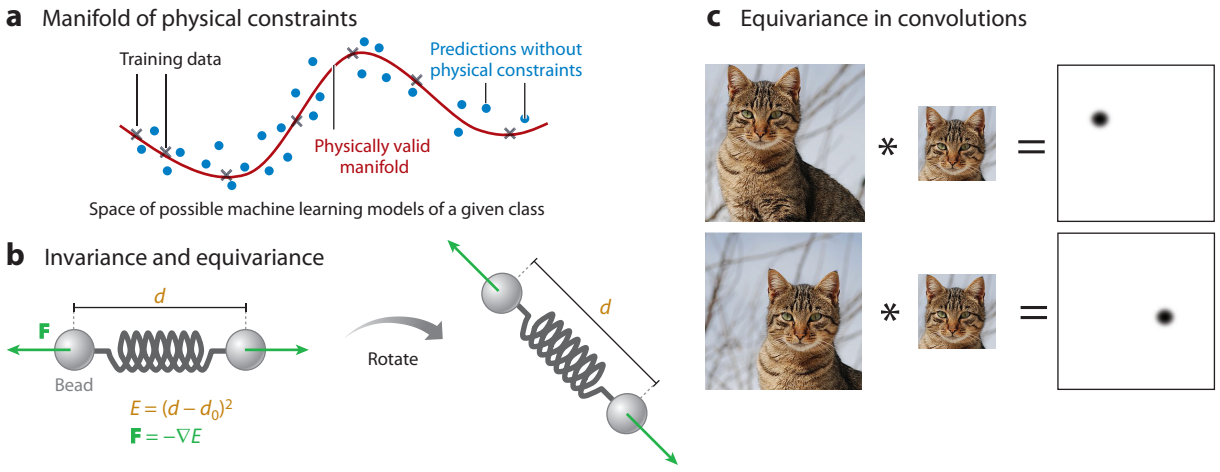

Figure 2

Illustration of physical constraints, invariance, and equivariance. (*a*) Physical constraints. These define a manifold of physically valid solutions for a given machine learning model class, such as a given neural network architecture: Only certain combinations of parameters will obey these physical constraints. Using data augmentation, the network can learn by heart to make almost physically valid predictions. Directly building physical constraints into the machine learning model is more data efficient and accurate, as then every prediction is physically meaningful. (*b*) Invariance and equivariance. Upon rotation of the beads, the spring energy is invariant, but the forces of the beads are equivariant; that is, they rotate in the same way. (*c*) Translational equivariance in convolutional layers. Convolving a translated image with a filter results in a translated feature map. Cat photograph in panel *c* reproduced from "Young Male Tabby Cat, Portugal" by Joaquim Alves Gaspar (**https://en.wikipedia.org/wiki/File:Cat_November_2010-1a.jpg**), used under CC BY 3.0.

but the force is equivariant as it rotates in the same way as $\mathbf{x}$ (**Figure 2*b***):

$$\begin{array}{ccc} \mathbf{x} & \xrightarrow{\mathbf{R}\cdot} & \mathbf{Rx} \\ \downarrow \nabla U(\cdot) & & \downarrow \nabla U(\cdot) \\ \nabla U(\mathbf{x}) & \xrightarrow{\mathbf{R}\cdot} & \mathbf{R}\nabla U(\mathbf{x}) = \nabla U(\mathbf{Rx}). \end{array}$$

Equivariances are closely linked to convolutions in ML. For example, standard convolutions are translation invariant: Each convolution kernel is a feature detector that is applied to each pixel neighborhood (99). When that feature is present in the image, the convolved image will show a signal in the corresponding position (**Figure 2*c***). When translating the input, the convolved image translates in the same way (**Figure 2*c***).

We briefly review below common invariances and equivariances useful for applications to molecular simulations and discuss strategies to incorporate them in an ML algorithm.

3.4.1. Rototranslational invariance and equivariance. Physical quantities that depend only on the interactions of the atoms within a molecule should be invariant with respect to translation $\mathbf{T}$ and rotation $\mathbf{R}\cdot$. Examples include potential and free energies of a molecule without an external field:

$$U(\mathbf{Rx} + \mathbf{T}) = U(\mathbf{x}).$$

The force is equivariant to rotation but invariant to translation (**Figure 2*b***; Section 3.4):

$$-\nabla U(\mathbf{Rx}+\mathbf{T}) = -\mathbf{R}\nabla U(\mathbf{x}).$$

Rototranslational invariance can be achieved by transforming the configuration $\mathbf{x}$ into rototranslationally invariant features, such as intramolecular distances or angles. Equivariance of the force can then be achieved by computing the force explicitly by a network layer that computes the derivatives with respect to $\mathbf{x}$, as it is done, for example, in SchNet (100) and CGnet (coarse-graining neural network) (20) (Section 4.3). Note that periodic systems, such as crystals and explicit-solvent boxes, have translational but not rotational invariances and equivariances.

3.4.2. Permutational invariance and equivariance. Physical quantities, such as QM energies, are invariant if we exchange the labels of identical atoms—for example, carbons. As the number of possible permutations increases exponentially with the number of identical particles, trying to learn permutation invariance by data augmentation is hopeless. Permutation invariance can be built into the ML model by choosing a functional form to compute the quantity of interest that is permutation invariant (see Reference 101 for the general conditions). Following the pioneering work in References 102–105, a specific choice that is common for networks that compute extensive quantities such as potential energy $U(\mathbf{x})$ is to model them as a sum,

$$U(\mathbf{x}) = \sum_i U_i(\mathbf{x}), \qquad 19.$$

where U_i is the contribution of the energy by the ith atom in its chemical environment (7, 11, 62, 100). To account for the multibody character of QM, $U_i(\mathbf{x})$ must generally also be a multibody function. Equation 19 can be implemented by using separate networks for the individual contributions U_i and adding up their results (**Figure 3**). The force resulting from Equation 19 is automatically permutation equivariant.

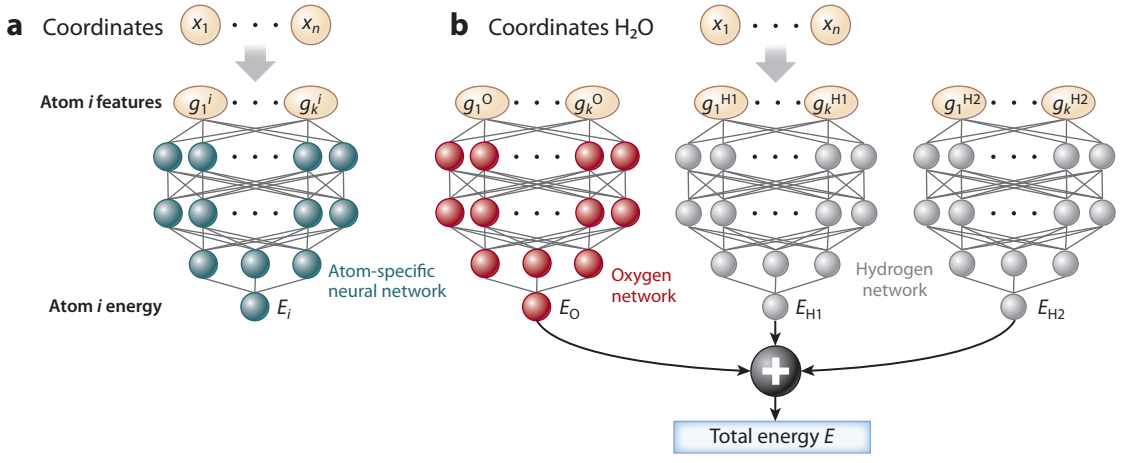

Figure 3

Behler–Parrinello networks for learning quantum energies. (*a*) A Behler–Parrinello network computes the atomic energy of a single atom i. The system coordinates are mapped to rototranslationally invariant features describing the chemical environment of atom i. (*b*) A molecular system such as H_2O is composed by employing a network copy for each atom. The parameters are shared between networks for same elements. The total energy is given by the sum of atomic energies, introducing permutation invariance.

Classical MD force fields define bonded interactions by assigning atoms to nodes in a bond graph, and thus, exchanging of individual atom pairs no longer preserves the energy. However, the energy is still invariant to the exchange of identical molecules, such as solvent, and it is important to take that into account for coarse graining (20) and generating samples from the equilibrium density (41).

A simple alternative to building permutation invariance into the ML function is to map all training and test data to a reference permutation. This can be done efficiently by so-called bipartite graph matching methods such as the Hungarian method (106) that are frequently used in recent learning models (10, 41, 107).

3.4.3. Energy conservation. Closed physical systems are energy conserving, which implies that the force field is defined by the gradient of the energy (Equation 3). When learning potential energy, it can be a great advantage to use force information, because each N-atom configuration is associated with only one energy but $3N$ forces, which may result in superior data efficiency when force labels are used during learning (10, 61). If we use supervised learning for coarse graining with thermodynamic consistency, we can only use forces, as no labels for the free energies are available (Section 2.3).

In these examples, we have labeled training data $[\mathbf{x}, \mathbf{f}(\mathbf{x})]_i$. Using a network that directly predicts the forces would not guarantee that Equation 3 holds. Therefore, when Equation 3 holds, it is common to learn an energy $U(\mathbf{x})$ and then compute the force in the network by using a gradient layer (20, 100) (Section 4.3). An alternative to ensure Equation 3 is gradient-domain ML (61).

3.4.4. Probability conservation and stochasticity. In statistical mechanics (both equilibrium and nonequilibrium), we are interested in the probability of events. An important principle is thus probability conservation—that is, the sum over all events is 1. A common approach to encode the probability of classes or events with a neural network is to use a SoftMax output layer, where the activation of each neuron can be defined as

$$y_i(\mathbf{u}) = \frac{e^{-u_i}}{\sum_j e^{-u_j}}, \qquad 20.$$

where j runs over all neurons in the layer. In this representation, $\mathbf{u}$ can be seen as a vector of energies giving rise to the Boltzmann probabilities $y_i(\mathbf{u})$; $y_i(\mathbf{u})$ are nonnegative, because of the exponential functions, and they sum up to 1.

In Markov state models of molecular kinetics, we would like to obtain a Markov transition matrix $\mathbf{P}$ that is stochastic; that is, $p_{ij} \geq 0$ for all elements, and $\sum_j p_{ij} = 1$ for all i. If the encoder $E(\mathbf{x})$ uses a SoftMax, then the estimator in Equation 14 will result in a transition matrix $\mathbf{P}$ that conserves probability ($\sum_j p_{ij} = 1$ for all i). SoftMax can be exploited in VAMPnets to simultaneously learn an embedding $E(\mathbf{x})$ from configurations to metastable states and a Markov transition matrix $\mathbf{P}$ (Section 4.4) (24).

3.4.5. Detailed balance. Detailed balance connects thermodynamics and dynamics. In a dynamical system that evolves purely in thermal equilibrium—that is, without applying external forces—the equilibrium distribution $\mu(\mathbf{x})$ and the transition probability $p(\mathbf{y} \mid \mathbf{x})$ are related by the following symmetry:

$$\mu(\mathbf{x})p(\mathbf{y} \mid \mathbf{x}) = \mu(\mathbf{y})p(\mathbf{x} \mid \mathbf{y}). \qquad 21.$$

Thus, the unconditional probabilities of forward and backward trajectories are equal. Enforcing Equation 21 in kinetic ML models ensures the spectral decomposition (Equation 11) is real valued, which is useful for many analyses.

To learn a kinetic model that obeys detailed balance, estimators other than Equation 14 must be used; these typically enforce the unconditional transition probabilities $p(\mathbf{x}, \mathbf{y}) = \mu(\mathbf{x})p(\mathbf{y} \mid \mathbf{x})$ to be symmetric (see Reference 90 for details).

3.5. Parameter Sharing and Convolutions

A decisive advance in the performance of neural networks for classical computer vision problems such as text or digit recognition came with going from fully connected dense networks to convolutional neural networks (99). Convolution layers are equivariant, thus helping with the detection of an object independent of its location (**Figure 2*c***), but the real efficiency of convolutional neural networks is due to parameter sharing.

In a classical dense network, all neurons of neighboring layers are connected and have independent parameters stored in a weight matrix $\mathbf{W}^l$. This leads to a problem with high-dimensional data. If we were to process relatively small images, say $100 \times 100 = 10^4$ pixels, and associate each pixel with a neuron, a single dense neural network layer l will have $10^4 \times 10^4 = 10^8$ parameters in $\mathbf{W}^l$. Not only would this be demanding in terms of memory and computing time but a network with so many independent parameters would also likely overfit and not be able to generalize to unknown data.

Convolutions massively reduce the number of independent parameters. A convolutional layer with a filter $\mathbf{w}$ acting on a one-dimensional signal $\mathbf{x}^{l-1}$ computes, before applying bias and nonlinearities,

$$z_i^l = \sum_j x_{i-j}^{l-1} w_j^l. \qquad 22.$$

In terms of an image, convolution applies the same filter $\mathbf{w}$ to every pixel neighborhood; in other words, all pixel transformations share the same parameters. Additionally, the filters $\mathbf{w}$ are usually much smaller than the signal $\mathbf{x}$ (often 3×3 for images), so each convolution has only a few parameters. For molecules, we extend this idea to continuous convolutions that use particle positions instead of pixels (Section 4.2).

Besides the sheer reduction of parameters, parameter sharing—for example, via convolution layers—is the keystone of transferability across chemical space. In convolutions of a grayscale image, we apply the same filters to every pixel, which implies translational equivariance but also means the same rules apply to all pixels. If we have multiple color channels, we have different channels in the filters as well, so pixels in the same color channels behave the same. In molecules we can apply the same idea to particle species. When one is learning energies from QM data, for example, every chemical element should be treated the same; that is, one should use the same convolution filters to sense its chemical environment. This treatment gives us a building-block principle that allows us to train on one set of molecules and make predictions for new molecules.

4. DEEP LEARNING ARCHITECTURES FOR MOLECULAR SIMULATION

In this section, we present specific methods and neural network architectures that have been proposed to tackle the ML problems discussed in Section 2.

4.1. Behler–Parrinello, Deep Potential Net, and ANI

Behler–Parrinello networks are one of the first applications of ML in the molecular sciences (7). They aim at learning and predicting potential energy surfaces from QM data, and they combine all of the relevant physical symmetries and parameter sharing for this problem (Section 3).

First, the molecular coordinates are mapped to rototranslationally invariant features for each atom i (**Figure 3*a***). In this step, the distances of neighboring atoms of a certain type, as well as the angles between two neighbors of certain types, are mapped to a fixed set of correlation functions that describe the chemical environment of atom i. These features are the input to a dense neural network that outputs one number, the energy of atom i in its environment. By design of the input feature functions, this energy is rototranslationally invariant. Parameters are shared between equivalent atoms—for example, all carbon atoms have the same network parameters for computing their atomic energies—but since the chemical environments will differ, their energies will differ. In a second step, the atomic energies are summed over all atoms of the molecule (**Figure 3*b***). This second step, combined with parameter sharing, achieves permutation invariance, as explained in Section 3.4.2. Transferability is achieved due to parameter sharing but also because the summation principle allows one to grow or shrink the network to molecules of any size, including sizes that were never seen in the training data.

A related approach is Deep Potential net (62), where each atom is treated in a local coordinate frame that has the rotational and translational degrees of freedom removed.

Behler–Parrinello networks are traditionally trained by energy matching (Section 2.1) (7, 56), but they can be trained with force matching if a gradient layer is added to compute the conservative force (Section 2.1; Equation 3). The Behler–Parrinello method has been further developed in the ANI network, for example, by extending it to more advanced functions involving two neighbors (11). While Behler–Parrinello networks have mainly been used to make predictions of the same molecular system in order to run MD simulations unaffordable by direct ab initio QM MD (56), ANI has been trained on density functional theory and coupled-cluster data across a large chemical space (11, 12, 108).

4.2. Deep Tensor Neural Networks, SchNet, and Continuous Convolutions

One of the first deep learning architectures to learn to represent molecules or materials was the family of deep tensor neural networks (DTNNs) (14), with its recent addition SchNet (58, 109). While in kernel-based learning methods (2, 110) chemical compounds are compared in terms of prespecified kernel functions (8, 111–113), DTNN and its extension SchNet learn a multiscale representation of the properties of molecules or materials from large data sets.

DTNNs were inspired by the language processing approach word-to-vec (114), where the role of a word within its grammatical or semantic context is learned and encoded in a parameter vector. Likewise, DTNNs learn a representation vector for each atom within its chemical environment (**Figure 4*b***, left). DTNN' s tensor construction algorithm then iteratively learns higher-order representations by first interacting with all pairwise neighbors, for example, extracting information implemented in the bond structure (**Figure 4*b***, middle). By stacking such interaction layers deep, DTNNs can represent the structure and statistics of multibody interactions in deeper layers. As DTNNs are end-to-end trained to predict certain QM quantities, such as potential energies, they learn the representation that is relevant for the task of predicting these quantities (115).

SchNet (58) uses a deep convolutional neural network (116, 117). Classically, convolutional neural networks were developed for computer vision using pixelated images, so they use discrete convolution filters. However, the particle positions of molecules cannot be discretized on a grid

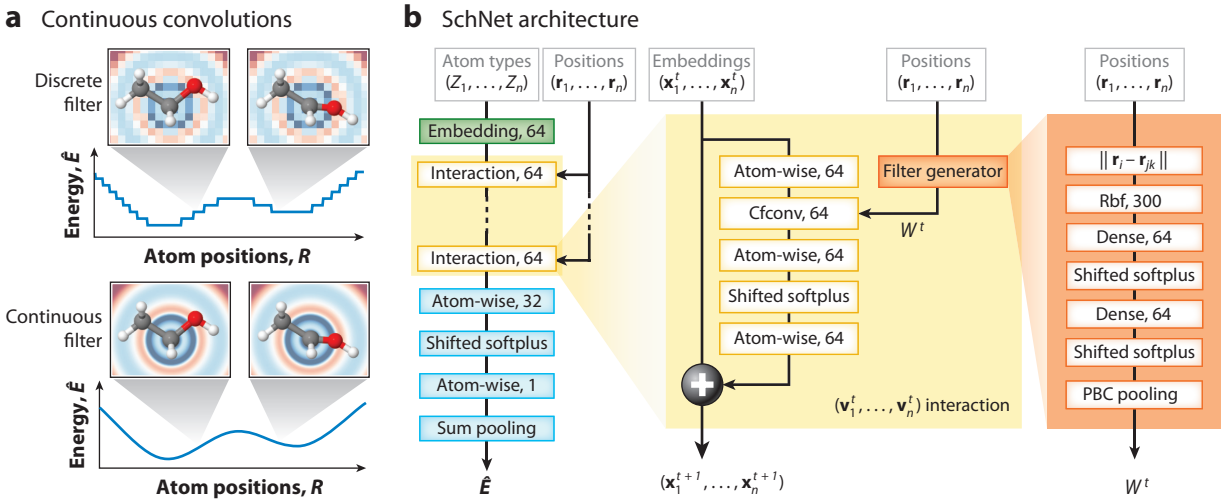

Figure 4

SchNet, a continuous convolution framework. (*a*) As molecular structures cannot be well discretized on a grid, SchNet generalizes the ConvNet approach to continuous convolutions between particles. (*b*, *left*) In the SchNet architecture, the input, consisting of atom types (chemical elements $Z_1, \ldots, Z_n$) and positions $\mathbf{r}_1, \ldots, \mathbf{r}_n$, is processed through several layers to produce atom-wise energies that are summed to a total energy. Numbers in boxes indicate network layer shapes (for details see 58, 108). (*b*, *middle*) The most important layer is the interaction layer, in which atoms interact via continuous convolution functions. The variable W^t denotes convolutional weights, and $\mathbf{v}$ are interactions. (*b*, *right*) Continuous convolutions are generated by dense neural networks that operate on interatomic distances, ensuring rototranslational invariance of the energy. Abbreviation: PBC, periodic boundary conditions. Figure adapted from Reference 100.

because QM properties such as the energy are highly sensitive to small position changes, such as the stretching of a covalent bond. For this reason, SchNet introduced continuous convolutions (58, 100), which are represented by filter-generating neural networks that map the rototranslationally invariant interatomic distances to the filter values used in the convolution (**Figure 4*b***, right).

DTNNs and SchNet have both reached highly competitive prediction quality both across chemical compound space and across configuration space in order to simulate MD. In addition to their prediction quality, their scalability to large data sets (109) and their ability to extract novel chemical insights by means of their learned representation make the DTNNs increasingly popular research tools.

4.3. Coarse Graining: CGnets

As mentioned in Section 2.3, ML has been used to define coarse-grained models for molecular systems. Both kernel methods (18) and deep neural networks (19, 20) have been designed to minimize the force-matching loss, Equation 9, for given coarse-graining mappings for specific systems.

In both cases, it has been shown that the incorporation of physical constraints is crucial to the success of the model. The training data are obtained by means of atomistic molecular dynamic simulations, and regions of the configurational space that are physically forbidden, such as configurations with broken covalent bonds or overlapping atoms, are not sampled and not included in the training. Without additional constraints, the machine cannot make predictions far away from the training data and thus will not reliably predict that the energy should diverge when approaching physically forbidden regions.

Excluding high-energy states such as broken bonds or colliding atoms is different from enforcing physical symmetries as described in Section 3.4. Rather, it is about enforcing the correct

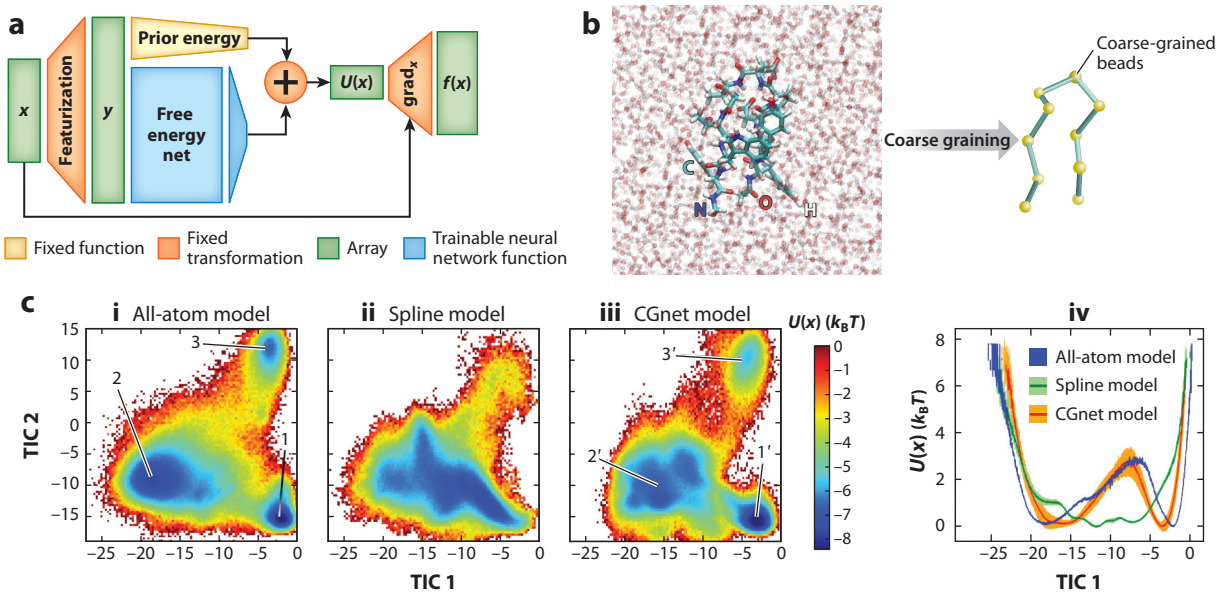

Figure 5

CGnet. (*a*) Using the CGnet neural network architecture to design a coarse-grained model by force matching, via the loss function of Equation 9. (*b*) Application of CGnet to the coarse graining of the miniprotein chignolin, from the fully atomistic and solved model to a C_α-only model of 10 beads. (*c*) Results of CGnet for chignolin. The term k_BT is thermal energy. (*i*) The free energy of the original atomistic model, as a function of the two reaction coordinates. State 1 is the folded state, state 2 is the unfolded state, and state 3 is a misfolded state. (*ii*) The free energy resulting from a coarse-grained model where only two-body terms are included in the energy function. (*iii*) The free energy resulting from CGnet. States 1′, 2′, and 3′ correspond to states 1, 2, and 3 in subpanel *i*. (*iv*) The same free energies as in subpanels *i–iii* but as a function of only one reaction coordinate. Abbreviations: CGnet, coarse-graining neural network; TIC, time-lagged independent component. Figure adapted from Reference 20.

asymptotic behavior of the energy when going toward an unphysical limit. CGnets proposed to achieve this by learning the difference from a simple prior energy that was defined to have the correct asymptotic behavior (20) (**Figure 5*a***). The exact form of this prior energy is not essential for success, as the CGnet can correct the prior energy where training data are available. In Reference 20, the prior energy consisted of harmonic terms for bonds and angles of coarse-grained particles whose equilibrium values and force constants were obtained with Boltzmann inversion, as well as excluded volume terms in the form of $(\sigma/r)^c$, where r is the interparticle distance and σ, c are hyperparameters.

As in Behler–Parrinello networks and SchNet, CGnet predicts a rototranslationally invariant energy as the first layer transforms the Cartesian coordinates into internal coordinates such as distances and angles (**Figure 5*a***). Furthermore, CGnet predicts a conservative and rotation-equivariant force field as the gradient of the total free energy with respect to input configuration **x** is computed self-consistently by the network (see **Figure 5*a***). The network is trained by minimizing the force-matching loss of this prediction (Equation 8).

Figure 5*b*,*c* shows an application of CGnets to the coarse graining of the miniprotein chignolin, in which all solvent molecules are coarse grained away and the atoms of each residue are mapped to the corresponding C_α atom (**Figure 5*b***). MD simulations performed with the force field predicted by the CGnet predict a free energy surface that is quantitatively similar to the free energy surface of the all-atom simulations, and they resolve the same metastable states (folded,

unfolded, and misfolded). In contrast, a spline-based coarse-grained model where only two-body terms are included in the energy function cannot reproduce the all-atom free energy surface and does not even predict that folded and unfolded are separated metastable states. These results clearly illustrate the importance of multibody interactions in the coarse-grained energy—for example, surface or volume terms that can describe implicit solvation. While the spline model can be dramatically improved by adding suitable terms to the list of features, this is not necessary when using a deep neural network that automatically learns the required multibody terms. Similar conclusions can be obtained by using Gaussian approximation potentials as the ML model to capture multibody terms in coarse-grained energy functions (18).

4.4. Kinetics: VAMPnets

VAMPnets (24) were introduced to replace the complicated and error-prone approach of constructing Markov state models by (*a*) searching for optimal features; (*b*) combining them into a low-dimensional representation $\mathbf{y}$, for example, via time-lagged independent component analysis (118); (*c*) clustering $\mathbf{y}$; (*d*) estimating the transition matrix $\mathbf{P}$; and (*e*) coarse graining $\mathbf{P}$. VAMPnets instead use a single end-to-end learning approach in which all of these steps are replaced by a deep neural network. This is possible because with the variational approach for conformational dynamics and VAMP principles (Section 2.4) (77, 89), loss functions are available that are suitable to train the embedding $E(\mathbf{x})$ and the propagator $\mathbf{P}$ simultaneously (see Section 2.4; Equation 13).

VAMPnets contain two network lobes representing the embedding $E(\mathbf{x})$. These networks transform the molecular configurations found at a time delay τ along the simulation trajectories (**Figure 6*a***). VAMPnets can be trained by minimizing the VAMP loss (Equation 15), which is meaningful for dynamics both with and without detailed balance (77). VAMPnets may, in general, use two distinct network lobes to encode the spectral representation of the left and right singular functions [which is important for nonstationary dynamics (119, 120)]. Extended dynamic mode decomposition with dictionary learning (121) uses an architecture similar to that of VAMPnets but is optimized by minimizing the regression error in latent space. To avoid collapsing to trivial embeddings, such as constant functions (see Section 2.4), a suitable regularization must be employed (121).

While hyperparameter selection can be performed by minimizing the variational loss (Equation 15) on a validation set (122–125), it is important to test the performance of a kinetic model on timescales beyond the training timescale τ. We can use the Chapman–Kolmogorov equation to test how well the learned model predicts longer times:

$$\mathbf{P}^n(\tau) \approx \mathbf{P}(n\tau). \qquad 23.$$

A common way to implement this test is to compare the leading eigenvalues $\lambda_i(\tau)$ of the left- and right-hand sides (81, 84, 126).

In Reference 24, parameters were shared between the VAMPnet nodes, and a unique embedding $E(\mathbf{x})$ was thereby learned. When detailed balance is enforced while computing $\mathbf{P}(\tau)$ (Equation 21), the loss function automatically becomes a variational approach for conformational dynamics score (89). In this case, the embedding $E(\mathbf{x})$ encodes the space of the dominant Markov operator eigenfunctions (24). This feature was extensively studied in state-free reversible VAMPnets (26).

To obtain a propagator that can be interpreted as a Markov state model, Mardt et al. (24) chose to use a SoftMax layer as an output layer, transforming the spectral representation to a soft indicator function similar to spectral clustering methods such as robust Perron cluster analysis

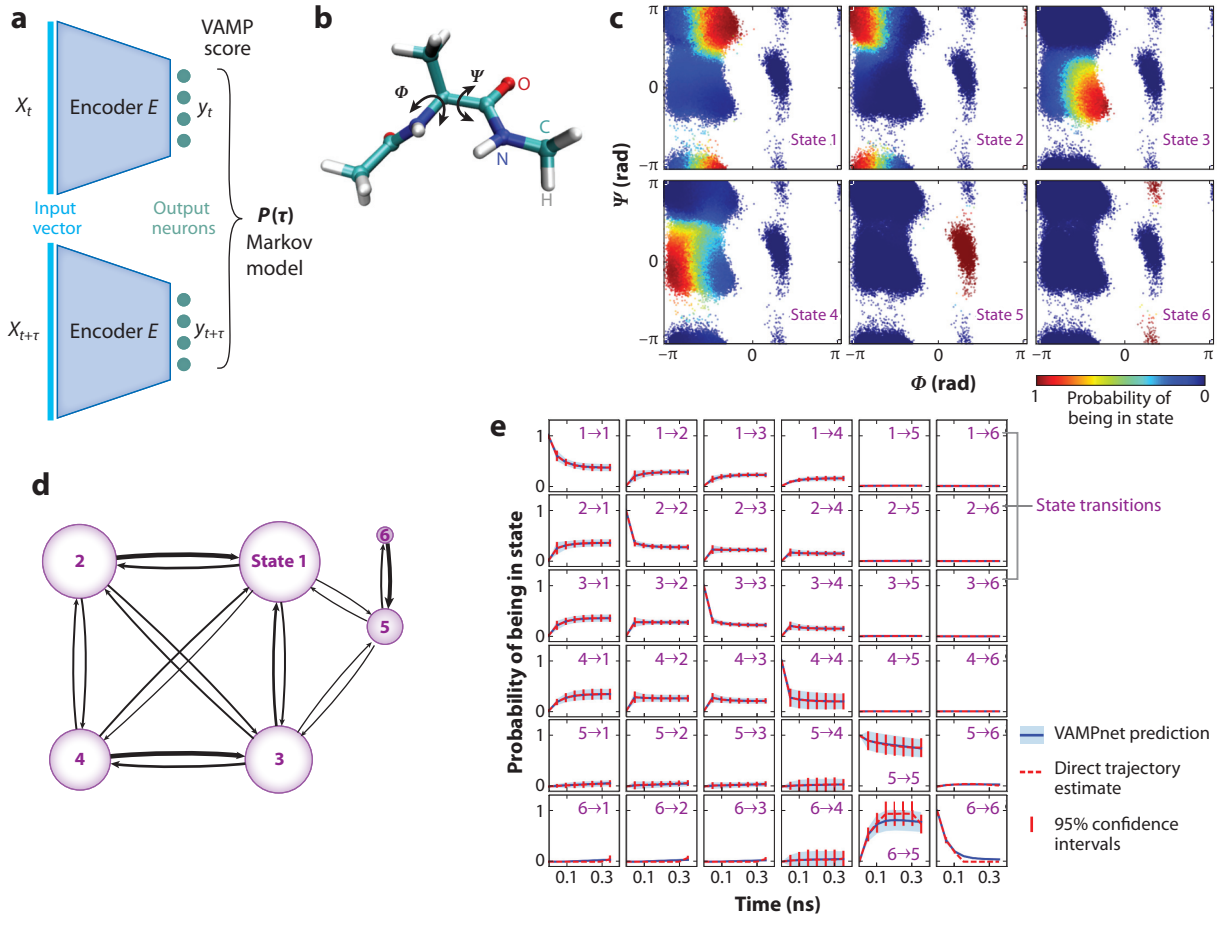

Figure 6

VAMPnet and application to alanine dipeptide. (*a*) A VAMPnet (24). The network includes an encoder E that transforms each molecular configuration $\mathbf{x}_t$ to a latent space of slow reaction coordinates $\mathbf{y}_t$ and is trained on pairs $(\mathbf{y}_t, \mathbf{y}_{t+\tau})$ sampled from the molecular dynamics simulation using the VAMP score (77). If the encoder performs a classification, the dynamical propagator $\mathbf{P}(\tau)$ is a Markov state model. (*b*) Structure of alanine dipeptide. The backbone torsion angles ϕ and ψ describe the slow kinetics of conformation changes, but Cartesian coordinates of heavy atoms are used as VAMPnet inputs here. (*c*) Classification of the VAMPnet encoder of molecular dynamics frames to metastable states in the ϕ and ψ space. Color corresponds to activation of the respective output neuron. (*d*) Equilibrium probabilities of Markov states (*purple circles*, with size proportional to probability; numbers are as in panels *c* and *e*) and transition probabilities given by $\mathbf{P}(\tau)$ (*arrows*, with thickness proportional to transition probability). (*e*) Chapman–Kolmogorov test comparing long-time predictions of the probability of being in a metastable state *j* at a certain time given a starting point in a metastable state *i* (*i*→*j* on each graph). The panel compares predictions by the VAMPnet model estimated at $\tau = 50$ ps with estimates at longer lag times. Figure adapted from Reference 24.

(127, 128). As a result, the propagator computed by Equation 14 conserves probability and is almost a transition matrix (Section 3.4.4), although it may have some negative elements with small absolute values.

The results described in Reference 24 (see, e.g., **Figure 6**) were competitive with and sometimes surpassed the state-of-the-art handcrafted Markov state model analysis pipeline. Given the rapid improvements in training efficiency and accuracy of deep neural networks seen in a broad

range of disciplines, we expect end-to-end learning approaches such as VAMPnets to dominate the field eventually.

In Reference 31, a deep generative Markov state model was proposed that, in addition to the encoder $E(\mathbf{x})$ and the propagator $\mathbf{P}$, learns a generative part that samples the conditional distribution of configurations in the next time step. The model can be operated in a recursive fashion to generate trajectories to predict the system evolution from a defined starting state and to propose new configurations. The deep generative Markov state model was demonstrated to provide accurate estimates of the long-time kinetics and to generate valid distributions for small MD benchmark systems.

4.5. Sampling/Thermodynamics: Boltzmann Generators

Boltzmann generators were introduced in Reference 41 to learn to sample equilibrium distributions (Equation 1). In contrast to standard generative learning, a Boltzmann generator does not attempt to learn the probability density from data but is trained to efficiently sample $\mu(\mathbf{x}) \propto \mathrm{e}^{-u(\mathbf{x})}$ using the dimensionless energy $u(\mathbf{x})$ as an input. A Boltzmann generator consists of two parts:

1. a generative model that is trained to propose samples from a probability distribution $p_X(\mathbf{x})$ that is similar to $\mu(\mathbf{x})$ and that allows us to evaluate $p_X(\mathbf{x})$ (up to a constant) for every $\mathbf{x}$, and
2. a reweighting procedure that takes proposals from $p_X(\mathbf{x})$ and generates unbiased samples from $\mu(\mathbf{x})$.

Boltzmann generators use a trainable generative network F_{zx} that maps latent space samples $\mathbf{z}$ from a simple prior—for example, a Gaussian normal distribution—to samples $\mathbf{x} \sim p_X(\mathbf{x})$. Training is done by combining the energy-based training using the KL divergence (Equation 17) and maximum likelihood (Equation 18).

For both training and reweighting, we need to be able to compute the probability $p_X(\mathbf{x})$ of generating a configuration $\mathbf{x}$. This can be achieved by the change-of-variables equation if F_{zx} is an invertible transformation (**Figure 7*a***) (94, 95). Such invertible networks are called flows due to the analogy of the transformed probability density with a fluid (94, 129). In Reference 41, the nonvolume preserving transformations RealNVP were employed (129), but the development of more powerful invertible network architectures is an active field of research (97, 130, 131). By stacking multiple invertible blocks, a deep invertible neural network is obtained that can encode a complex transformation of variables.

Figure 7*b–e* illustrates the Boltzmann generator on a condensed matter model system that contains a bistable dimer in a box densely filled with repulsive solvent particles (**Figure 7*b***). Opening or closing the dimer is a rare event, and it involves the collective rearrangement of the solvent particles due to the high particle density. Using short MD simulation in the open and closed states as an initialization, the Boltzmann generator can be trained to sample open, closed, and previously unseen transition states by generating Gaussian random variables in latent space (**Figure 7*c***) and feeding them through the transformation F_{zx}. Such samples have realistic structures and close-to-equilibrium energies (**Figure 7*d***). Free energy differences can be computed by employing reweighting (**Figure 7*e***). There is a direct relationship between the temperature of the canonical ensemble and the variance of the latent-space Gaussian of the Boltzmann generator (41). This allows us to learn to generate thermodynamics, such as the temperature-dependent free energy profiles, using a single Boltzmann generator (**Figure 7*e***). Finally, as the latent space concentrates configurations of equilibrium probability around the origin, Boltzmann generators can be used to generate physically realistic reaction pathways by performing linear interpolations in latent space.

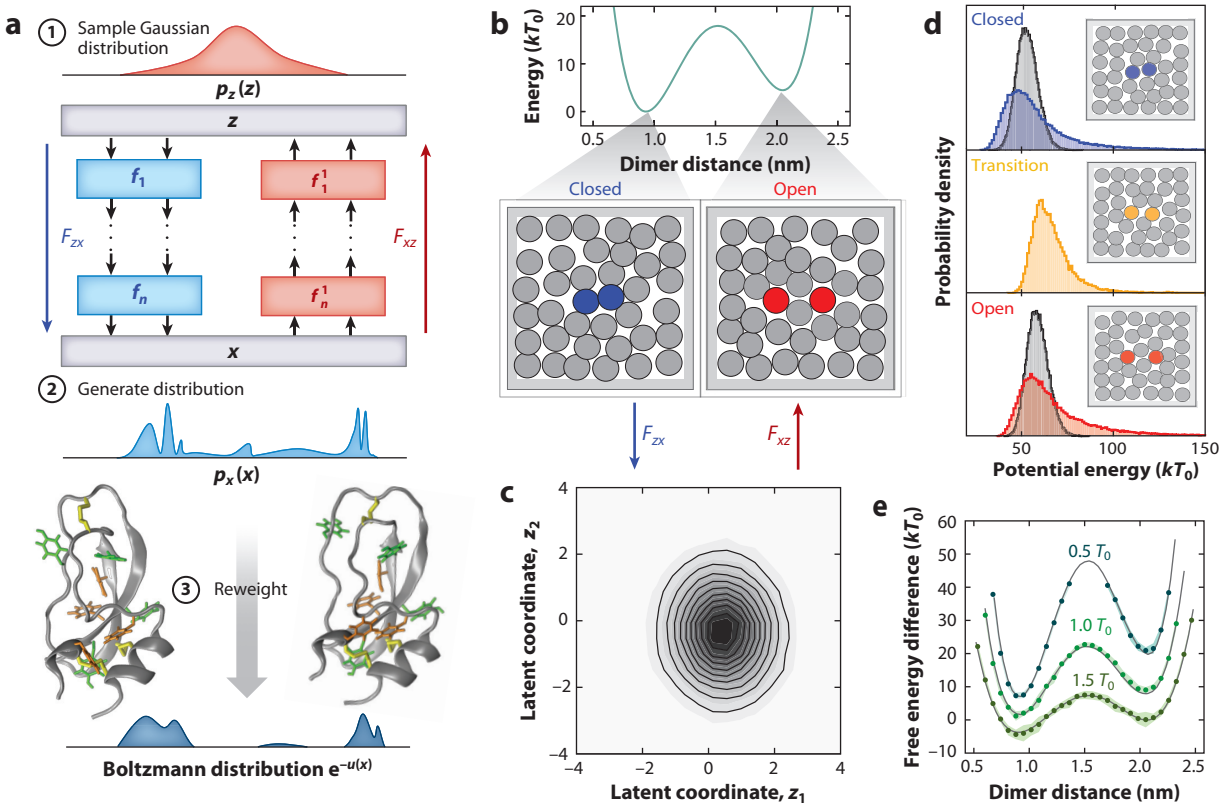

Figure 7

Boltzmann generators. (*a*) A Boltzmann generator is trained by minimizing the difference between its generated distribution and the desired Boltzmann distribution. Generation proceeds by drawing latent space samples **z** from a simple prior distribution (e.g., Gaussian) and transforming them to configurations **x** via an invertible flow: a deep neural network F_{zx} and its inverse, F_{xz}. To compute thermodynamics, such as configurational free energies, the samples must be reweighted to the Boltzmann distribution. (*b*) Repulsive particle system with bistable dimer. Closed (*blue*) and open (*red*) configurations are from molecular dynamics simulations (input data). (*c*) Distribution of molecular dynamics simulation data in latent space coordinates z_1, z_2 after training the Boltzmann generator. (*d*) Potential energy distribution from molecular dynamics (*gray*) and the Boltzmann generator for closed (*blue*), open (*red*), and transition (*yellow*) configurations. Insets show one-shot Boltzmann generator samples. (*e*) Free energy differences as a function of dimer distance and relative temperature sampled with Boltzmann generators (*green circles*) and umbrella sampling (*gray lines*). The shaded areas represent the 68% confidence intervals. Figure adapted from Reference 41.

5. DISCUSSION

Despite rapid advances in the field of ML for molecular simulation, there are still significant open problems that need to be addressed, in all the areas discussed above.

5.1. Accuracy and Efficiency in Quantum Chemical Energies and Forces

To be practically useful, an ML model for both PES and atomic forces is needed that (*a*) can yield accuracy of 0.2–0.3 kcal/mol for the energy per functional group and about 1 kcal/(mol·Å) for the force per atom, (*b*) is not much more expensive to evaluate than classical force fields, (*c*) scales to large molecules such as proteins, and (*d*) is transferable to different covalent and noncovalent environments. Such a universal model does not exist yet.

Crucial steps toward items *a* and *b* have been recently taken by symmetrized gradient-domain machine learning (sGDML), a kernel-based approach to constructing molecular force fields (10, 61, 132, 133). Currently, sGDML already enables MD simulations with electrons and nuclei treated at essentially exact QM level for molecules with up to 20–30 atoms.

Network-based approaches, such as SchNet and ANI, are better suited to items *c* and *d*, as they break down the energy in local interactions of atoms with their environment, enabling a building-block principle that is by design better scalable to molecules of different sizes and transferable across chemical space. However, these approaches do not reach the high accuracy in configuration space that sGDML does. Combining high accuracy in configuration and chemical space remains an active research topic.

5.2. Long-Ranged Interactions

The vast majority of approaches to make ML inference on molecular structures are based on local chemical information. Current neural networks for modeling molecular energies use the summation principle (e.g., Equation 19) to sum up local energies $E_i(\mathbf{x})$ of atom i with its neighbors. While multibody and long-ranged energies can be obtained by stacking multiple layers (14, 58, 100)—the working principle of deep convolution networks (116)—there are fundamental physical limits of this approach: Long-ranged interactions such as electrostatics cannot be cut off.

For classical point-charge models, long-ranged electrostatics methods have been developed, such as the Ewald summation method for periodic systems (134). One option is to combine short-ranged ML models with such long-ranged electrostatics methods. To avoid double counting interactions, one must also predict atomic charges, which is an active field of research (135, 136). An alternative option—currently unexplored territory—is to develop neural network structures for particle interactions that can compute long-ranged interactions by design.

In addition to electrostatics, van der Waals dispersion interactions can also have a substantial long-range character; that is, they can extend to separations of tens of nanometers or more in large molecular and nanoscale systems (137–139). Developing ML models that correctly treat the QM many-body nature of van der Waals interactions remains a difficult challenge to overcome (15).

5.3. Quantum Kinetics

With the availability of chemically transferable ML models that have quantum chemical accuracy, the next open problem is to sample metastable states and long-timescale kinetics. Although available ML models for predicting QM energies and forces are still significantly slower than force fields, the vast array of enhanced sampling methods and kinetic models (Sections 2.4 and 4.4) will likely allow us to explore the kinetics of quantum chemical systems on timescales of microseconds and beyond. A plethora of new physical insights that we cannot access with current MD force fields await us there. For example, what is the role of protonation dynamics in mediating protein folding or function?

5.4. Transferability of Coarse-Grained Models

An outstanding question in the design of coarse-grained models is that of transferability across chemical space. Bottom-up coarse-grained models are useful in practice if they can be parameterized on small molecules and then used to predict the dynamics of systems much larger than what is possible to simulate with atomistic resolution. It is not clear to what extent transferability of coarse-grained models can be achieved and how that depends on the coarse-graining mapping

(140, 141). Compared with the manual design of few-body free energy functionals, ML free energies can help with transferability, as they are able to learn the important multibody effects—for example, to model neglected solvent molecules implicitly (Section 4.3) (18–20).

It is natural to consider Behler–Parrinello networks or SchNet as a starting point for modeling transferable coarse-grained energies, but their application is nontrivial: It is a priori unclear what the interacting particles are in the coarse-grained model and how to define their types, as they are no longer given by the chemical element. Furthermore, these networks assume permutation invariance between identical particles, while classical MD force fields do not have permutation invariance of atoms within the same molecule. Therefore, particle network structures that can handle bonding graphs need to be developed.

5.5. Kinetics of Coarse-Grained Models

While coarse-grained MD models may perform well in reproducing the thermodynamics of the atomistic system, they may fail in reproducing the kinetics. Existing approaches include adding fictitious particles (142) or training the coarse-grained model with spectral matching (143). There are indications that the kinetics can be approximated up to a global scaling factor in barrier-crossing problems when the barriers are well approximated (144), which could be achieved by identifying the slow reaction coordinates (63) and assigning more weight to the transition state in force matching or relative entropy minimization. This area of research is still underdeveloped.

5.6. Transferable Prediction of Intensive Properties

Extensive properties such as potential energies can be well predicted across chemical space, as they can be conceptually broken down as a sum of parts that can be learned separately. This is not possible with intensive properties such as spectra or kinetics, and for this reason, the prediction of such properties is, as yet, far behind.

5.7. Equivariant Generative Networks with Parameter Sharing

Generative networks, such as Boltzmann generators (Section 4.5), have been demonstrated to be able to generate physically realistic one-shot samples of model systems and proteins in implicit solvent (41). To scale to larger systems, we will need to build the invariances of the energy, such as the exchange of identical solvent particles, into the transformation and to include parameter sharing (Section 3.5), such that we can go beyond just sampling the probability density of one given system with energy $u(\mathbf{x})$ and instead generalize from a data set of examples of one class of molecules, such as solvated proteins. To this end, equivariant networks with parameter sharing need to be developed for generative learning; these are, to date, not available.

5.8. Explainable Artificial Intelligence

Recently, the increasing popularity of explainable artificial intelligence methods (see, e.g., 145–148) has allowed us to gain insight into the inner workings of deep learning algorithms. In this manner, it has become possible to extract how a problem is solved by the deep model. This allows, for example, for the detection of so-called clever Hans solutions (147)—that is, nonsensical solutions relying on artifactual or nonphysical aspects in data. Combined with networks that learn a representation, such as DTNN/SchNet (14, 100) and VAMPnets (24), these inspection methods may provide scientific insights into the mechanisms that give rise to the predicted physicochemical quantity and thereby fuel the development of new theories.

DISCLOSURE STATEMENT

The authors are not aware of any affiliations, memberships, funding, or financial holdings that might be perceived as affecting the objectivity of this review.

ACKNOWLEDGMENTS

We gratefully acknowledge funding from the European Commission (ERC CoG 772230 "ScaleCell" to F.N. and ERC CoG grant BeStMo to A.T.), Deutsche Forschungsgemeinschaft (CRC1114/A04 to F.N.; EXC 2046/1, project ID 390685689, to K.-R.M.; and GRK2433 DAEDALUS to F.N. and K.-R.M.), the MATH+ Berlin Mathematics research center (AA1-6 to F.N. and EF1-2 to F.N. and K.-R.M.), Einstein Foundation Berlin (Einstein Visiting Fellowship to C.C.), the National Science Foundation (grants CHE-1265929, CHE-1740990, CHE-1900374, and PHY-1427654 to C.C.), the Welch Foundation (grant C-1570 to C.C.), an Institute for Information and Communications Technology Planning and Evaluation grant funded by the Korean government (2017-0-00451 and 2017-0-01779 to K.-R.M.), and the German Ministry for Education and Research (grants 01IS14013A-E, 01GQ1115, and 01GQ0850 to K.-R.M.). We thank Stefan Chmiela and Kristof Schütt for help with **Figures 1** and **5**.

LITERATURE CITED

1. Dirac PAM. 1929. Quantum mechanics of many-electron systems. *Proc. R. Soc. A* 123:714–33
2. Müller KR, Mika S, Rätsch G, Tsuda K, Schölkopf B. 2001. An introduction to kernel-based learning algorithms. *IEEE Trans. Neural Netw.* 12:181–201
3. Vapnik VN. 1999. An overview of statistical learning theory. *IEEE Trans. Neural Netw.* 10:988–99
4. Bishop CM. 2006. *Pattern Recognition and Machine Learning*. Singapore: Springer
5. LeCun Y, Bengio Y, Hinton G. 2015. Deep learning. *Nature* 521:436–44
6. Goodfellow I, Bengio Y, Courville A. 2016. *Deep Learning*. Cambridge, MA: MIT Press
7. Behler J, Parrinello M. 2007. Generalized neural-network representation of high-dimensional potential-energy surfaces. *Phys. Rev. Lett.* 98:146401
8. Rupp M, Tkatchenko A, Müller KR, Lilienfeld OAV. 2012. Fast and accurate modeling of molecular atomization energies with machine learning. *Phys. Rev. Lett.* 108:058301
9. Bartók AP, Payne MC, Kondor R, Csányi G. 2010. Gaussian approximation potentials: the accuracy of quantum mechanics, without the electrons. *Phys. Rev. Lett.* 104:136403
10. Chmiela S, Sauceda HE, Müller KR, Tkatchenko A. 2018. Towards exact molecular dynamics simulations with machine-learned force fields. *Nat. Commun.* 9:3887
11. Smith JS, Isayev O, Roitberg AE. 2017a. ANI-1: an extensible neural network potential with DFT accuracy at force field computational cost. *Chem. Sci.* 8:3192–203
12. Smith JS, Nebgen BT, Zubatyuk R, Lubbers N, Devereux C, et al. 2018. Outsmarting quantum chemistry through transfer learning. ChemRxiv 6744440. **https://doi.org/10.26434/chemrxiv.6744440.v1**
13. Brockherde F, Vogt L, Li L, Tuckerman ME, Burke K, Müller KR. 2017. Bypassing the Kohn–Sham equations with machine learning. *Nat. Commun.* 8:872
14. Schütt KT, Arbabzadah F, Chmiela S, Müller KR, Tkatchenko A. 2017. Quantum-chemical insights from deep tensor neural networks. *Nat. Commun.* 8:13890
15. Bereau T, DiStasio RA Jr., Tkatchenko A, von Lilienfeld OA. 2018. Non-covalent interactions across organic and biological subsets of chemical space: physics-based potentials parametrized from machine learning. *J. Chem. Phys.* 148:241706
16. Welborn M, Cheng L, Miller TF III. 2018. Transferability in machine learning for electronic structure via the molecular orbital basis. *J. Chem. Theory Comput.* 14:4772–79
17. Cheng L, Welborn M, Christensen AS, Miller TF III. 2019. A universal density matrix functional from molecular orbital-based machine learning: transferability across organic molecules. *J. Chem. Phys.* 150:131103

18. John ST, Csányi G. 2017. Many-body coarse-grained interactions using Gaussian approximation potentials. *J. Phys. Chem. B* 121:10934–49
19. Zhang L, Han J, Wang H, Car R, Weinan E. 2018. DeePCG: constructing coarse-grained models via deep neural networks. *J. Chem. Phys.* 149:034101
20. Wang J, Olsson S, Wehmeyer C, Pérez A, Charron NE, et al. 2019. Machine learning of coarse-grained molecular dynamics force fields. *ACS Cent. Sci.* 5:755–67
21. Durumeric AEP, Voth GA. 2019. Adversarial-residual-coarse-graining: applying machine learning theory to systematic molecular coarse-graining. *J. Chem. Phys.* 151:124110
22. Wehmeyer C, Noé F. 2018. Time-lagged autoencoders: deep learning of slow collective variables for molecular kinetics. *J. Chem. Phys.* 148:241703
23. Hernández CX, Wayment-Steele HK, Sultan MM, Husic BE, Pande VS. 2018. Variational encoding of complex dynamics. *Phys. Rev. E* 97:062412
24. Mardt A, Pasquali L, Wu H, Noé F. 2018. VAMPnets: deep learning of molecular kinetics. *Nat. Commun.* 9:5
25. Ribeiro JML, Bravo P, Wang Y, Tiwary P. 2018. Reweighted autoencoded variational Bayes for enhanced sampling (RAVE). *J. Chem. Phys.* 149:072301
26. Chen W, Sidky H, Ferguson AL. 2019. Nonlinear discovery of slow molecular modes using state-free reversible VAMPnets. *J. Chem. Phys.* 150:214114
27. Jung H, Covino R, Hummer G. 2019. Artificial intelligence assists discovery of reaction coordinates and mechanisms from molecular dynamics simulations. arXiv:1901.04595 [physics.chem-ph]
28. Stecher T, Bernstein N, Csányi G. 2014. Free energy surface reconstruction from umbrella samples using Gaussian process regression. *J. Chem. Theory Comput.* 10:4079–97
29. Mones L, Bernstein N, Csányi G. 2016. Exploration, sampling, and reconstruction of free energy surfaces with Gaussian process regression. *J. Chem. Theory Comput.* 12:5100–10
30. Schneider E, Dai L, Topper RQ, Drechsel-Grau C, Tuckerman ME. 2017. Stochastic neural network approach for learning high-dimensional free energy surfaces. *Phys. Rev. Lett.* 119:150601
31. Wu H, Mardt A, Pasquali L, Noé F. 2018a. Deep generative Markov state models. *Adv. Neural Inf. Process. Syst.* 31:3975–84
32. Olsson S, Noé F. 2019. Dynamic graphical models of molecular kinetics. *PNAS* 116:15001–6
33. Valsson O, Parrinello M. 2014. Variational approach to enhanced sampling and free energy calculations. *Phys. Rev. Lett.* 113:090601
34. Bonati L, Zhang YY, Parrinello M. 2019. Neural networks-based variationally enhanced sampling. *PNAS* 116:17641–47
35. Zhang J, Yang YI, Noé F. 2019. Targeted adversarial learning optimized sampling. *J. Phys. Chem. Lett.* 10:5791–97
36. McCarty J, Parrinello M. 2017. A variational conformational dynamics approach to the selection of collective variables in metadynamics. *J. Chem. Phys.* 147:204109
37. Sultan MM, Pande VS. 2017. tICA-metadynamics: accelerating metadynamics by using kinetically selected collective variables. *J. Chem. Theory Comput.* 13:2440–47
38. Doerr S, Fabritiis GD. 2014. On-the-fly learning and sampling of ligand binding by high-throughput molecular simulations. *J. Chem. Theory Comput.* 10:2064–69
39. Zimmerman MI, Bowman GR. 2015. FAST conformational searches by balancing exploration/exploitation trade-offs. *J. Chem. Theory Comput.* 11:5747–57
40. Plattner N, Doerr S, Fabritiis GD, Noé F. 2017. Protein-protein association and binding mechanism resolved in atomic detail. *Nat. Chem.* 9:1005–11
41. Noé F, Olsson S, Köhler J, Wu H. 2019. Boltzmann generators—sampling equilibrium states of many-body systems with deep learning. *Science* 365:eaaw1147
42. Jiménez J, Skalic M, Martinez-Rosell G, Fabritiis GD. 2018. K_{DEEP}: protein–ligand absolute binding affinity prediction via 3D-convolutional neural networks. *J. Chem. Inf. Model.* 58:287–96
43. Skalic M, Varela-Rial A, Jiménez J, Martínez-Rosell G, Fabritiis GD. 2018. LigVoxel: inpainting binding pockets using 3D-convolutional neural networks. *Bioinformatics* 35:243–50

44. Wu Z, Ramsundar B, Feinberg EN, Gomes J, Geniesse C, et al. 2018b. MoleculeNet: a benchmark for molecular machine learning. *Chem. Sci.* 9:513–30
45. Feinberg EN, Sur D, Wu Z, Husic BE, Mai H, et al. 2018. PotentialNet for molecular property prediction. *ACS Cent. Sci.* 4:1520–30
46. Gómez-Bombarelli R, Wei JN, Duvenaud D, Hernández-Lobato JM, Sánchez-Lengeling B, et al. 2018. Automatic chemical design using a data-driven continuous representation of molecules. *ACS Cent. Sci.* 4:268–76
47. Popova M, Isayev O, Tropsha A. 2018. Deep reinforcement learning for de novo drug design. *Sci. Adv.* 4:eaap7885
48. Winter R, Montanari F, Steffen A, Briem H, Noé F, Clevert DA. 2019a. Efficient multi-objective molecular optimization in a continuous latent space. *Chem. Sci.* 10:8016–24
49. Winter R, Montanari F, Noé F, Clevert DA. 2019b. Learning continuous and data-driven molecular descriptors by translating equivalent chemical representations. *Chem. Sci.* 10:1692–701
50. Butler KT, Davies DW, Cartwright H, Isayev O, Walsh A. 2018. Machine learning for molecular and materials science. *Nature* 559:547–55
51. Sanchez-Lengeling B, Aspuru-Guzik A. 2018. Inverse molecular design using machine learning: generative models for matter engineering. *Science* 361:360–65
52. Lindorff-Larsen K, Maragakis P, Piana S, Eastwood MP, Dror RO, Shaw DE. 2012. Systematic validation of protein force fields against experimental data. *PLOS ONE* 7:e32131
53. Nerenberg PS, Jo B, So C, Tripathy A, Head-Gordon T. 2012. Optimizing solute-water van der Waals interactions to reproduce solvation free energies. *J. Phys. Chem. B* 116:4524–34
54. Huang J, Rauscher S, Nawrocki G, Ran T, Feig M, et al. 2016. CHARMM36m: an improved force field for folded and intrinsically disordered proteins. *Nat. Methods* 14:71–73
55. Robustelli P, Piana S, Shaw DE. 2018. Developing a molecular dynamics force field for both folded and disordered protein states. *PNAS* 115:E4758–66
56. Behler J. 2016. Perspective: machine learning potentials for atomistic simulations. *J. Chem. Phys.* 145:170901
57. Li Z, Kermode JR, Vita AD. 2015. Molecular dynamics with on-the-fly machine learning of quantum-mechanical forces. *Phys. Rev. Lett.* 114:96405
58. Schütt KT, Sauceda HE, Kindermans PJ, Tkatchenko A, Müller KR. 2018. SchNet—a deep learning architecture for molecules and materials. *J. Chem. Phys.* 148:241722
59. Gastegger M, Behler J, Marquetand P. 2017. Machine learning molecular dynamics for the simulation of infrared spectra. *Chem. Sci.* 8:6924–35
60. Dral PO, Owens A, Yurchenko SN, Thiel W. 2017. Structure-based sampling and self-correcting machine learning for accurate calculations of potential energy surfaces and vibrational levels. *J. Chem. Phys.* 146:244108
61. Chmiela S, Tkatchenko A, Sauceda HE, Poltavsky I, Schütt KT, Müller KR. 2017. Machine learning of accurate energy-conserving molecular force fields. *Sci. Adv.* 3:e1603015
62. Han J, Zhang L, Car R, Weinan E. 2018. Deep Potential: a general representation of a many-body potential energy surface. *Phys. Rev. Lett.* 120:143001
63. Noé F, Clementi C. 2017. Collective variables for the study of long-time kinetics from molecular trajectories: theory and methods. *Curr. Opin. Struct. Biol.* 43:141–47
64. Galvelis R, Sugita Y. 2017. Neural network and nearest neighbor algorithms for enhancing sampling of molecular dynamics. *J. Chem. Theory Comput.* 13:2489–500
65. Sidky H, Whitmer JK. 2018. Learning free energy landscapes using artificial neural networks. *J. Chem. Phys.* 148:104111
66. Ribeiro JML, Tiwary P. 2018. Toward achieving efficient and accurate ligand-protein unbinding with deep learning and molecular dynamics through RAVE. *J. Chem. Theory Comput.* 15:708–19
67. Chen W, Ferguson AL. 2018. Molecular enhanced sampling with autoencoders: on-the-fly collective variable discovery and accelerated free energy landscape exploration. *J. Comput. Chem.* 39:2079–102

68. Sultan MM, Wayment-Steele HK, Pande VS. 2018. Transferable neural networks for enhanced sampling of protein dynamics. *J. Chem. Theory Comput.* 14:1887–94
69. Guo AZ, Sevgen E, Sidky H, Whitmer JK, Hubbell JA, de Pablo JJ. 2018. Adaptive enhanced sampling by force-biasing using neural networks. *J. Chem. Phys.* 148:134108
70. Noid WG, Chu JW, Ayton GS, Krishna V, Izvekov S, et al. 2008. The multiscale coarse-graining method. I. A rigorous bridge between atomistic and coarse-grained models. *J. Chem. Phys.* 128:244114
71. Noid WG. 2013. Perspective: coarse-grained models for biomolecular systems. *J. Chem. Phys.* 139:090901
72. Ciccotti G, Lelièvre T, Vanden-Eijnden E. 2008. Projection of diffusions on submanifolds: application to mean force computation. *Commun. Pure Appl. Math.* 61:371–408
73. Clementi C. 2008. Coarse-grained models of protein folding: toy-models or predictive tools? *Curr. Opin. Struct. Biol.* 18:10–15
74. Boninsegna L, Banisch R, Clementi C. 2018. A data-driven perspective on the hierarchical assembly of molecular structures. *J. Chem. Theory Comput.* 14:453–60
75. Wang W, Gómez-Bombarelli R. 2018. Variational coarse-graining for molecular dynamics. arXiv:1812.02706 [physics.chem-ph]
76. Sarich M, Noé F, Schütte C. 2010. On the approximation quality of Markov state models. *Multiscale Model. Simul.* 8:1154–77
77. Wu H, Noé F. 2017. Variational approach for learning Markov processes from time series data. arXiv:1707.04659 [stat.ML]
78. Buchete NV, Hummer G. 2008. Coarse master equations for peptide folding dynamics. *J. Phys. Chem. B* 112:6057–69
79. Noé F, Doose S, Daidone I, Löllmann M, Chodera JD, et al. 2011. Dynamical fingerprints for probing individual relaxation processes in biomolecular dynamics with simulations and kinetic experiments. *PNAS* 108:4822–27
80. Schütte C, Fischer A, Huisinga W, Deuflhard P. 1999. A direct approach to conformational dynamics based on hybrid Monte Carlo. *J. Comput. Phys.* 151:146–68
81. Swope WC, Pitera JW, Suits F. 2004. Describing protein folding kinetics by molecular dynamics simulations: 1. Theory. *J. Phys. Chem. B* 108:6571–81
82. Noé F, Horenko I, Schütte C, Smith JC. 2007. Hierarchical analysis of conformational dynamics in biomolecules: transition networks of metastable states. *J. Chem. Phys.* 126:155102
83. Chodera JD, Dill KA, Singhal N, Pande VS, Swope WC, Pitera JW. 2007. Automatic discovery of metastable states for the construction of Markov models of macromolecular conformational dynamics. *J. Chem. Phys.* 126:155101
84. Prinz JH, Wu H, Sarich M, Keller BG, Senne M, et al. 2011. Markov models of molecular kinetics: generation and validation. *J. Chem. Phys.* 134:174105
85. Mezić I. 2005. Spectral properties of dynamical systems, model reduction and decompositions. *Nonlinear Dyn.* 41:309–25
86. Schmid PJ. 2010. Dynamic mode decomposition of numerical and experimental data. *J. Fluid Mech.* 656:5–28
87. Tu JH, Rowley CW, Luchtenburg DM, Brunton SL, Kutz JN. 2014. On dynamic mode decomposition: theory and applications. *J. Comput. Dyn.* 1:391–421
88. Wu H, Nüske F, Paul F, Klus S, Koltai P, Noé F. 2017. Variational Koopman models: slow collective variables and molecular kinetics from short off-equilibrium simulations. *J. Chem. Phys.* 146:154104
89. Noé F, Nüske F. 2013. A variational approach to modeling slow processes in stochastic dynamical systems. *Multiscale Model. Simul.* 11:635–55
90. Noé F. 2018. Machine learning for molecular dynamics on long timescales. arXiv:1812.07669 [physics.chem-ph]
91. Smolensky P. 1986. Information processing in dynamical systems: foundations of harmony theory. In *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, Vol. 1: *Foundations*, ed. DE Rumelhart, JL McLelland, pp. 194–281. Cambridge, MA: MIT Press
92. Kingma DP, Welling M. 2014. Auto-encoding variational Bayes. arXiv:1312.6114 [stat.ML]

93. Goodfellow I, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, et al. 2014. Generative adversarial networks. *Adv. Neural Inf. Process. Syst.* 27:2672–80
94. Dinh L, Krueger D, Bengio Y. 2015. NICE: Non-linear Independent Components Estimation. arXiv:1410.8516 [cs.LG]
95. Tabak EG, Vanden-Eijnden E. 2010. Density estimation by dual ascent of the log-likelihood. *Commun. Math. Sci.* 8:217–33
96. Karras T, Aila T, Laine S, Lehtinen J. 2018. Progressive growing of GANs for improved quality, stability, and variation. arXiv:1710.10196 [cs.NE]
97. Kingma DP, Dhariwal P. 2018. Glow: generative flow with invertible 1×1 convolutions. arXiv:1807:03039 [stat.ML]
98. van den Oord A, Li Y, Babuschkin I, Simonyan K, Vinyals O, et al. 2018. Parallel WaveNet: fast high-fidelity speech synthesis. *Proc. Mach. Learn. Res.* 80:3918–26
99. LeCun Y, Bottou L, Bengio Y, Haffner P. 1998a. Gradient-based learning applied to document recognition. *Proc. IEEE* 86:2278–324
100. Schütt K, Kindermans PJ, Sauceda HE, Chmiela S, Tkatchenko A, Müller KR. 2017. SchNet: a continuous-filter convolutional neural network for modeling quantum interactions. *Adv. Neural Inf. Process. Syst.* 30:991–1001
101. Zaheer M, Kottur S, Ravanbakhsh S, Poczos B, Salakhutdinov RR, Smola AJ. 2017. Deep sets. *Adv. Neural Inf. Process. Syst.* 30:3391–401
102. Ercolessi F, Tosatti E, Parrinello M. 1986. Au (100) surface reconstruction. *Phys. Rev. Lett.* 57:719
103. Tersoff J. 1986. New empirical model for the structural properties of silicon. *Phys. Rev. Lett.* 56:632
104. Ferrante J, Smith JR, Rose JH. 1983. Diatomic molecules and metallic adhesion, cohesion, and chemisorption: a single binding-energy relation. *Phys. Rev. Lett.* 50:1385
105. Abell GC. 1985. Empirical chemical pseudopotential theory of molecular and metallic bonding. *Phys. Rev. B* 31:6184
106. Kuhn HW. 1955. The Hungarian method for the assignment problem. *Nav. Res. Logist. Q.* 2:83–97
107. Reinhard F, Grubmüller H. 2007. Estimation of absolute solvent and solvation shell entropies via permutation reduction. *J. Chem. Phys.* 126:014102
108. Smith JS, Isayev O, Roitberg AE. 2017b. ANI-1, a data set of 20 million calculated off-equilibrium conformations for organic molecules. *Sci. Data* 4:170193
109. Schütt KT, Kessel P, Gastegger M, Nicoli KA, Tkatchenko A, Müller KR. 2019. SchNetPack: a deep learning toolbox for atomistic systems. *J. Chem. Theory Comput.* 15:448–55
110. Vapnik V. 1995. *The Nature of Statistical Learning Theory*. New York: Springer
111. Hansen K, Montavon G, Biegler F, Fazli S, Rupp M, et al. 2013. Assessment and validation of machine learning methods for predicting molecular atomization energies. *J. Chem. Theory Comput.* 9:3404–19
112. Hansen K, Biegler F, Ramakrishnan R, Pronobis W, Von Lilienfeld OA, et al. 2015. Machine learning predictions of molecular properties: accurate many-body potentials and nonlocality in chemical space. *J. Phys. Chem. Lett.* 6:2326–31
113. Bartók AP, Kondor R, Csányi G. 2013. On representing chemical environments. *Phys. Rev. B* 87:184115
114. Mikolov T, Sutskever I, Chen K, Corrado GS, Dean J. 2013. Distributed representations of words and phrases and their compositionality. *Adv. Neural Inf. Process. Syst.* 26:3111–19
115. Braun ML, Buhmann JM, Müller KR. 2008. On relevant dimensions in kernel feature spaces. *J. Mach. Learn. Res.* 9:1875–906
116. LeCun Y, Boser B, Denker JS, Henderson D, Howard RE, et al. 1989. Backpropagation applied to handwritten zip code recognition. *Neural Comput.* 1:541–51
117. LeCun Y, Bottou L, Orr GB, Müller KR. 1998b. Efficient backprop. In *Neural Networks: Tricks of the Trade*, ed. JB Orr, KR Müller, pp. 9–53. Berlin: Springer
118. Perez-Hernandez G, Paul F, Giorgino T, De Fabritiis G, Noé F. 2013. Identification of slow molecular order parameters for Markov model construction. *J. Chem. Phys.* 139:015102
119. Koltai P, Ciccotti G, Schütte C. 2016. On metastability and Markov state models for non-stationary molecular dynamics. *J. Chem. Phys.* 145:174103

120. Koltai P, Wu H, Noé F, Schütte C. 2018. Optimal data-driven estimation of generalized Markov state models for non-equilibrium dynamics. *Computation* 6:22
121. Li Q, Dietrich F, Bollt EM, Kevrekidis IG. 2017. Extended dynamic mode decomposition with dictionary learning: a data-driven adaptive spectral decomposition of the Koopman operator. *Chaos* 27:103111
122. Bießmann F, Meinecke FC, Gretton A, Rauch A, Rainer G, et al. 2010. Temporal kernel CCA and its application in multimodal neuronal data analysis. *Mach. Learn.* 79:5–27
123. McGibbon RT, Pande VS. 2015. Variational cross-validation of slow dynamical modes in molecular kinetics. *J. Chem. Phys.* 142:124105
124. Husic BE, McGibbon RT, Sultan MM, Pande VS. 2016. Optimized parameter selection reveals trends in Markov state models for protein folding. *J. Chem. Phys.* 145:194103
125. Scherer MK, Husic BE, Hoffmann M, Paul F, Wu H, Noé F. 2019. Variational selection of features for molecular kinetics. *J. Chem. Phys.* 150:194108
126. Noé F, Schütte C, Vanden-Eijnden E, Reich L, Weikl TR. 2009. Constructing the full ensemble of folding pathways from short off-equilibrium simulations. *PNAS* 106:19011–16
127. Deuflhard P, Weber M. 2005. Robust Perron cluster analysis in conformation dynamics. *Linear Algebra Appl.* 398:61–84
128. Röblitz S, Weber M. 2013. Fuzzy spectral clustering by PCCA+: application to Markov state models and data classification. *Adv. Data Anal. Classif.* 7:147–79
129. Dinh L, Sohl-Dickstein J, Bengio S. 2016. Density estimation using real NVP. arXiv:1605.08803 [cs.LG]
130. Rezende DJ, Mohamed S. 2015. Variational inference with normalizing flows. arXiv:1505.05770 [stat.ML]
131. Grathwohl W, Chen RTQ, Bettencourt J, Sutskever I, Duvenaud D. 2018. FFJORD: free-form continuous dynamics for scalable reversible generative models. arXiv:1810.01367 [cs.LG]
132. Sauceda HE, Chmiela S, Poltavsky I, Müller KR, Tkatchenko A. 2019. Molecular force fields with gradient-domain machine learning: construction and application to dynamics of small molecules with coupled cluster forces. *J. Chem. Phys.* 150:114102
133. Chmiela S, Sauceda HE, Poltavsky I, Müller KR, Tkatchenko A. 2019. sGDML: constructing accurate and data efficient molecular force fields using machine learning. *Comput. Phys. Commun.* 240:38
134. Darden T, Perera L, Li L, Pedersen L. 1999. New tricks for modelers from the crystallography toolkit: the particle mesh Ewald algorithm and its use in nucleic acid simulations. *Structure* 7:55–60
135. Unke OT, Meuwli M. 2019. PhysNet: a neural network for predicting energies, forces, dipole moments and partial charges. *J. Chem. Theory Comput.* 15:3678–93
136. Nebgen B, Lubbers N, Smith JS, Sifain AE, Lokhov A, et al. 2018. Transferable dynamic molecular charge assignment using deep neural networks. *J. Chem. Theory Comput.* 14:4687–98
137. Ambrosetti A, Ferri N, DiStasio RA Jr., Tkatchenko A. 2016. Wavelike charge density fluctuations and van der Waals interactions at the nanoscale. *Science* 351:1171–76
138. Hermann J, DiStasio AR Jr., Tkatchenko A. 2017. First-principles models for van der Waals interactions in molecules and materials: concepts, theory, and applications. *Chem. Rev.* 117:4714–58
139. Stoehr M, Van Voorhis T, Tkatchenko A. 2019. Theory and practice of modeling van der Waals interactions in electronic-structure calculations. *Chem. Soc. Rev.* 48:4118–54
140. Mullinax JW, Noid WG. 2009. Extended ensemble approach for deriving transferable coarse-grained potentials. *J. Chem. Phys.* 131:104110
141. Thorpe IF, Goldenberg DP, Voth GA. 2011. Exploration of transferability in multiscale coarse-grained peptide models. *J. Phys. Chem. B* 115:11911–26
142. Davtyan A, Voth GA, Andersen HC. 2016. Dynamic force matching: construction of dynamic coarse-grained models with realistic short time dynamics and accurate long time dynamics. *J. Chem. Phys.* 145:224107
143. Nüske F, Boninsegna L, Clementi C. 2019. Coarse-graining molecular systems by spectral matching. *J. Chem. Phys.* 151:044116
144. Bereau T, Rudzinski JF. 2018. Accurate structure-based coarse graining leads to consistent barrier-crossing dynamics. *Phys. Rev. Lett.* 121:256002

145. Bach S, Binder A, Montavon G, Klauschen F, Müller KR, Samek W. 2015. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PLOS ONE* 10:e0130140
146. Montavon G, Samek W, Müller KR. 2018. Methods for interpreting and understanding deep neural networks. *Digit. Signal Process.* 73:1–15
147. Lapuschkin S, Wäldchen S, Binder A, Montavon G, Samek W, Müller KR. 2019. Unmasking Clever Hans predictors and assessing what machines really learn. *Nat. Commun.* 10:1096
148. Samek W, Montavon G, Vedaldi A, Hansen LK, Müller KR, eds. 2019. *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*. Cham, Switz.: Springer