Machine Learning for the Construction of Force Fields

David Zimmer

May 17, 2021

Institute for Computational Physics, University of Stuttgart

1 Introduction

Nowadays, computer simulations have become a standard tool in essentially all fields of chemistry, condensed matter physics, and materials science. An accurate description of the atomic interactions is of vital importance for carrying out reliable computer simulations [12]. The most prominent approach to describe those interactions are electronic structure methods and classical force fields. Electronic structure methods like density function theory and coupled cluster methods deliver excellent accuracy but are limited to small systems of a few hundred atoms and to timescales of a few picoseconds [5]. Force fields on the other side are capable to describe larger system for a longer period of time but fail when it comes to a simple description of bond breaking and phase transitions. In order to keep up with state-of-the-art experiments and the ever growing complexity of the investigated problems, there is a constantly increasing need for simulations of more realistic, i.e. larger, model systems with improved accuracy. One promising approach are machine learned models that can achieve near *ab initio* accuracy with a computational complexity $\mathcal{O}(N)$ where N is the number of atoms in the system [7]. This handout shall give a brief overview of the state of the art of machine learned potentials while also pointing out the difficulties, strengths and limits of the method.
First ,an overview on machine learning will be given followed by a description of the two most dominant approaches for machine learned potentials (ML-Potentials). After that a closer look will be taken on the concept of data and data processing. Last but no least limits and future outlooks on machine learned potentials will be elaborated.

2 Machine Learning

In the most general form machine learning algorithms are algorithms that learn with the data they are handed. The larger the dataset, the more accurate their statements and predictions become. Hereby, the field of machine learning splits in three different primary learning models [6]:

- Supervised Learning
- Unsupervised Learning
- Reinforcement Learning

To create machine learned potentials, one enters the field of supervised learning which itself can be divided into the area of *classification* problems and *regression* problems. Classification problems are mainly found in image recognition while regression is used to find overall trends in data such as price prediction or in this case energy predictions. There are several ways of performing these calculations. In this work, only two of such approaches will be described, which are Gaussian Process Regression and Neural Networks. The underlying idea of both methods is the same. They predict an output $\widetilde{y}$ for an yet unknown input $\widetilde{\boldsymbol{x}}$ based on a given set of n previous observations $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^{n}$ [11]. This set is called *training data*. In the case of fitting energy surfaces, the input might be the coordinates of a atomic configuration and the output is the potential energy of that configuration.

2.1 Gaussian Approximation Potentials

When deriving an energy surface using a Gaussian Process, one speaks of a Gaussian Approximation Potential (GAP). The basic idea of regression is finding a relation between an input vector $\boldsymbol{x}$ and an output y. For instance a linear relation like $\beta\boldsymbol{x} = y$. When knowing β, one can calculate $\widetilde{y}$ for an arbitrary input $\widetilde{\boldsymbol{x}}$. In practice this is not so easy since a linear approximation of the energy surface will not be sufficient. In the nonlinear case the problem is more in finding similarities between different inputs in order to map them on similar energies. This is done by a *kernel* function. There is a wide variety of possible kernel functions. A typical choice is the Gaussian kernel (or squared exponential kernel) [12]

$$k(\boldsymbol{x}, \boldsymbol{y}) = \sigma \exp\left(-\frac{\|\boldsymbol{x} - \boldsymbol{y}\|_2^2}{2l^2}\right). \tag{1}$$

σ and l are hyperparameters describing the amplitude and the length scale on which different inputs correlate. Using the Gaussian Kernel, two vectors that are far from each other will be mapped on 0 and two input coordinates, that are close to each other will be mapped on 1. This corresponds to a similarity measurement. Computing all the similarities of input pairs leads to the kernel matrix [1]

$$K_{ij} = k(x_i, x_j) \tag{2}$$

which is for a set of n observations an $n \times n$ matrix.
Before coming to the actual Gaussian process, it is now helpful to recap some features of a *multivariant Gaussian distribution* [11]. The probability density of such a distribution can be expressed by

$$p(\boldsymbol{x}) \sim \exp\Big(-\frac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu})^T \Sigma^{-1} \cdot (\boldsymbol{x} - \boldsymbol{\mu})\Big). \tag{3}$$

$\boldsymbol{x}$ is a vector valued stochastic variable, $\boldsymbol{\mu}$ the mean of this distribution and Σ^{-1} the covariance matrix. To understand a multivariant Gaussian consider the two dimensional case, pictured in Figure 1. The density shown in the middle panel the case, where Σ^{-1} is the identity matrix and x_1 and x_2 are uncorrelated. In the two other cases, the off diagonal is nonzero and x_1 and x_2 are correlated. One can say that the off diagonal elements describe the correlation between the variables. In the case of a Gaussian Process this covariance matrix is given by the kernel matrix. Consider now a sets of observations $\{(\boldsymbol{x}_i, f_i)\}_{i=1}^m$, called *training* set and a new input vector $\boldsymbol{x_*}$ for which one is interested in the energy. The distribution of this energy output can be considered as a joint distribution of the training set energies and the new energy. The Gaussian process now provides a distribution of functions, that when evaluated at an input of the training data, give the associated energy output. In addition to that, the mean energy output for the new input $\boldsymbol{x_*}$ can also be calculated by first computing the covariance vector using the kernel function

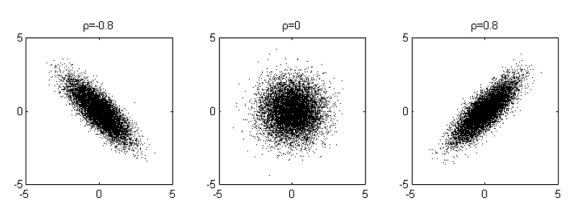

Figure 1: Plot of 10000 samples of a two dimensional Gaussian distribution for different correlation coefficients ρ. Left: $\rho = -0.8$, middle: $\rho = -0$ and right: $\rho = 0.8$.

$$k_{i,*} = k(\boldsymbol{x}_i, \boldsymbol{x}_*) \tag{4}$$

and then by

$$\bar{\boldsymbol{f}}_* = \boldsymbol{k}_*^T K^{-1} \boldsymbol{f}. \tag{5}$$

K is the kernel matrix of the training data. $\bar{\boldsymbol{f}}_*$ is then taken as the energy prediction of an arbitrary input $\boldsymbol{x}_*$ and $K^{-1}\boldsymbol{f}$ is in some sense the Gaussian Approximation Potential.[11] It is important to state here, that first inverting K is an $\mathcal{O}(n^3)$ operation that is only relevant during training. The computation of the energy prediction is an $\mathcal{O}(n)$ operation but it is important to emphasize here that n is not the system size but again the training size, that can be significantly bigger than the system size.

2.2 Neuronal Network Potentials

An alternative approach to GAP are Neuronal Network Potentials (NNPs). They can consist of one more artificial neuronal networks (NNs). NNs are *universal approximators* that enable in principle arbitrary accurate approximation of unknown multidimensional functions based on a set of known function values.
In principle NNPs can be defined by three aspects [4]:

- They provide direct functional relation between the atomic configuration and the potential energy.
- They use a set of data obtained from a single electronic structure method.
- They do not contain any approximation apart from the intrinsic limitations of the chosen electronic structure method.

The underlying concept of an NNP is a neuronal network. In figure 2 such a neuronal network is shown. The final *neuron* or *node* is the output node and is the final energy. In the first *layer*, the input layer, the configuration in form of a coordinate input $\boldsymbol{G} = \{G_i\}$ is handed to the NN. The number of layer determines the mathematical flexibility of the network and normally each layer contains ~ 50 nodes. Each node is connected to all nodes in the following layer by a *weight* $a_{i,j}^{k,l}$, connecting the i^{th} node of the k^{th} layer with the j^{th} node of the l^{th} layer. Additionally each node is connected to a bias node by an according bias weight b_i^k that connects the bias node with the i^{th} node of the k^{th} layer. The value of the node i in layer k is then calculated with

$$y_i^k = f_i^k\Big(b_i^k + \sum_{j=1}^{N_{k-1}} a_{j,i}^{k-1,k} \cdot y_j^{k-1}\Big) \qquad (6)$$

which is essentially a linear combination of the values of nodes of the previous layer plus the bias. The weights are fitted during the training. To fit also nonlinear functions (e.g. the PES) a nonlinear *activation function* f_i^k is applied. A typical choice is hyperbolic tangent or sigmoid function as both have a nonlinear region and approach 1 for large values and either -1 or 0 for negative values. The choice of activation function should be differentiable with respect to its inputs such that the functional expression of the energy is differentiable as well. For the NNP shown in figure 2 the energy written in its closed form is

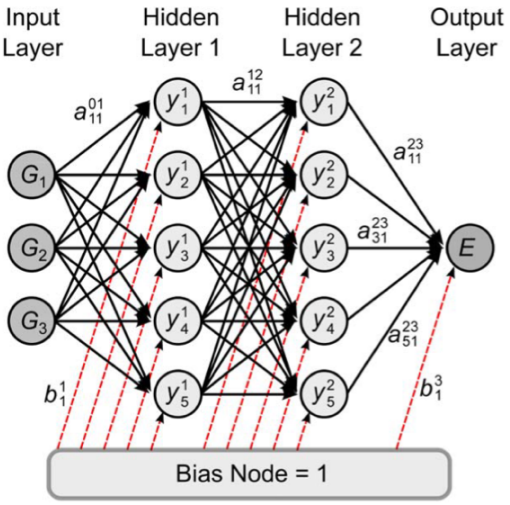

Figure 2: Schematic structure of a 3-5-5-1 single feed forward NN. The chain of numbers describes how many nodes are in each layer and characterises the NN. Goal is to give a functional relation between the input vector $\boldsymbol{G}$ and the energy output [4].

$$E = f_1^3\Big(b_1^3 + \sum_{k=1}^{5} a_{k1}^{23} \cdot f_k^2\Big(b_k^2 + \sum_{j=1}^{5} a_{jk}^{12} \cdot f_j^1\Big(b_j^1 + \sum_{i=1}^{3} a_{ij}^0 1 \cdot G_i\Big)\Big)\Big). \qquad (7)$$

In some sense a neural network can therefore simply be interpreted as a function which parameters are fitted during training. Still, it is considered a *nonparametric* fitting procedure. This is due to the fact that the parameters are not associated with fixed energy contributions and physical relations. During the training the weights are fitted in such a way that a given error function that compares energy predictions with reference energies, is minimized. In practice such single feed forward networks are not used. They are not symmetric under permutation of atoms. To overcome this problem one uses so called *high dimensional* neural networks. Instead of calculating the total energy of the system, they focus on the local energy of each atom by using a single feed forward NN for each individual atom. The total energy is then computed as the sum of the local energies [4].

3 Data

As previously mentioned, a machine learning algorithm is one that learns with the data it is handed. The following section gives a brief overview on where that data comes from, how to reduce the data necessary for teaching an ML algorithm, and most importantly, how to present the data before handing it to the machine.
In general all data used in an ML process can be expressed as a set of n observations $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^{n}$ which assign an output y_i to every input $\boldsymbol{x}_i$. In most cases one differs between the training data and the test data. As the title suggests the training data is that which the algorithm is trained on. The test data is then used to make sure the training is sufficient.

3.1 Data Sources

In the case of ML potentials, the data fitted is mostly the result of electronic structure calculations such as DFT, applied to different configurations of the system of interest. Data from more complex methods can also be used here, like Hartree-Fock or coupled cluster calculations. In principal, one could also use Classical Force Field data to fit those. When using data from a certain structure method, one should stick to that method and not include data from others as this would influence the fit significantly and would provide bad results. It is often the case that the configurations are drawn from long and computationally expensive ab-initio MD trajectories. Another approach is to run a classical MD simulation, draw samples from that and then use a electronic structure method to calculate the correct energy of those configurations [2].

3.2 Data Selection

The huge amount of computationally expensive data necessary to get reliable results from a machine learning algorithm is one of the biggest obstacles of ML potentials. Another significant problem is to make sure that the training data describes configuration space sufficiently. That means one has to make sure that as many unique atomic environments as possible are contained in the training data. Although it may sound simple, in practice it might be difficult to decide whether a given configuration is unique or not. It is also not clear what makes an atomic environment unique. To find those unique configuration a different number of sampling methods can be used. Possible methods are for instance [7]:

- a random selection over the full MD trajectory
- a global energy selection, selecting uniformly across the energy values
- an atomic energy selection, sampling configurations uniformly based on atomic energies
- a force selection in which the net force on the i^{th} atom is used to indicate a unique environment.

A comparison of the different approaches is shown in Figure 3. Hereafter different errors of the resulting MD simulations for data-sets constructed with different sampling methods are shown. The results suggest that sampling uniformly over the atomic energy distribution leads to more accurate and robust machine learned potentials than sampling the same amount of training data from distributions of global properties.

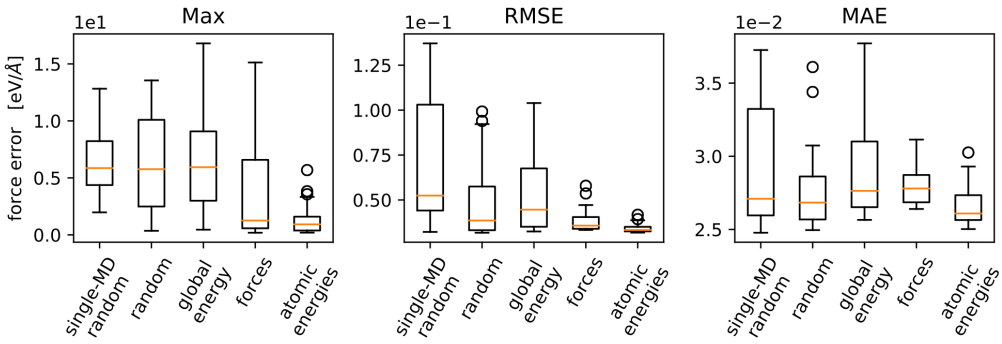

Figure 3: Comparison of maximum error *left*, root-mean-square error (RMSE) *middle*, and the mean absolute error (MAE) *right* of force prediction of the same ML model trained on data-sets sampled with different methods [7].

3.3 Descriptors

One key component of machine learning is the so called *sample efficiency*, describing how much data is necessary to train the algorithm sufficiently. The higher the efficiency the less data is needed for a certain degree of accuracy. As explained in the section before, one way to improve the sampling efficiency is to select the right training data. Another way is to use the correct description. A simple example to understand this is when looking at a water molecule. When representing the positions of the three atoms in Cartesian coordinates an ML algorithm can learn the energy associated with a certain configuration. When rotating this configuration around the oxygen atom, the coordinates of the hydrogen atoms will change. Although the potential of that configuration did not change compared to the previous, an ML potential will not recognize that configuration as known and therefore predicts it energy output wrongly[3]. One way to overcome this problem is, to include all rotational variances of a configuration in the training set. This would not be feasible. Another approach is to change the representation of the data in a way that both of those configurations are represented in the same way. This is done by using so called *descriptors*. Further to rotational invariance, there are a handful of requirements a descriptor has to fulfill.

3.3.1 Requirements

The first and perhaps most obvious criteria for a descriptor is invariance under the fundamental symmetry transformation (*i.*), under which the energy is also invariant as well. These are translation, rotation, reflection and permutation of atoms of the same species. A descriptor must map different atomic environments onto different representations. This is called uniquness (*ii.*). Another requirement is being continuous and in the best case differentiable (*iii.*). Non continuous descriptors lead to a steep increase of computation power. Using differentiable descriptors leads to a differentiable potential energy which is necessary to derive forces for a MD simulation. Although it is not a necessary requirement it is favorable that a descriptor is computationally efficient (*iv.*). A crucial requirement is that the resulting structure of the descriptor should be suitable for regression (*v.*). Last but not least every local environment should be encodable in a descriptor. This is feature is called generality (*vi.*). This means also that when applying the descriptor to another system, one should not have to build it from scratch [3].

In practice representations do not satisfy all six requirements and the choice also depends on the type and amount of data. Often the fulfillment of the above criteria depend on so called hyperparameters that are specific parameters defining a descriptor.

3.3.2 Descriptor Example

There are two primary strategies to deal with previously introduced invariances. The first one can use an invariant k-body function like it is done with the *symmetry functions* method (SF) and the *Coulomb matrix* (CM) [14]. The second approach is to explicitly symmetrize a function as it is done in the *many body tensor representation* (MBTR) or the *smooth overlap of atomic positions* method (SOAP) [10]. Here, only the last of those four will be described further. Additional information about the others can be found in the cited sources.
The smooth overlap of atom positions method (SOAP) expands a central atoms local neighborhood density approximated by Gaussian functions located at the position of atoms in orthogonal radial and spherical harmonics. The, at first, more intuitive approach of describing the density in terms of delta functions would lead to a non differantiable representation. The spherical harmonic expansion reads

$$\rho(\boldsymbol{r}) = \sum_{n,l,m} c_{n,l,m}\, g_n(\boldsymbol{r})\, Y_{l,m}(\boldsymbol{r}). \tag{8}$$

The radial harmonics g_n can be replaced whith alternative basis functions like Gaussian function or polynomials. $Y_{l,m}$ are the spherical harmonics and $c_{n,l,m}$ are the expansion coefficients. These coefficients are now used then used to calculate the power spectrum

$$p_{n,n',l} = \sum_{m} c_{n,l,m}\, c*_{n*,l,m}\,. \tag{9}$$

Alternatively, one can compute the bispectrum as well. Both leads to a representation invariant under rotation and permutation. While garanteeing this invariance, the computation of power- or bispectrum is equivalent to evaluating a kernel function. The maximum number of radial and angular basis functions, the broadening width of the Gaussians and the cutoff radius are numerical hyperparameters that need to be determined before using SOAP for computation.

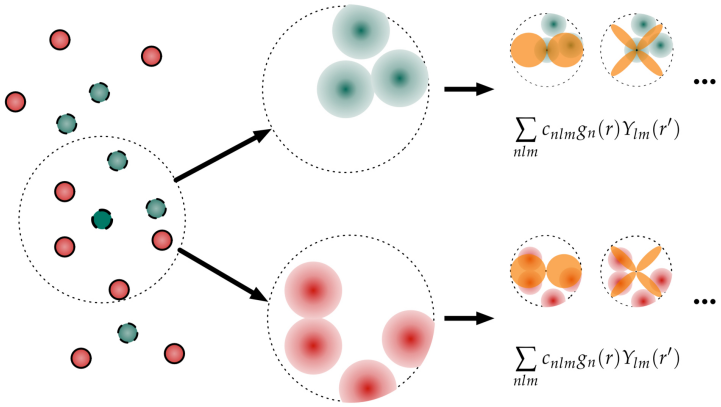

Figure 4: Schematic figure of describing the local environment containing two different atomic species [10].

In the end, it is always a question tied to the system of interest, which descriptor is suited best. The performance of different descriptors depends on the system- and data size and type of data [2].

4 Results

As the ML potential is always a fit of the underlying electronic structure method the reference for an observable derived from a MD simulation is always the value of that observable calculated by the associated *ab intio* MD simulation. A machine learned potential is better, the smaller the difference between *ab initio* result and ML-MD is. Since during training the energy is fitted, it is obvious that the first observable one would compare is the energy and since one is interested in molecular dynamics the error in forces should also be as small as possible. In Figure 5 such a plot is shown. One can see that the output of the GAP are in good accordance with the DFT energies and forces [8]. Besides reproducing the energy correctly, ML potentials

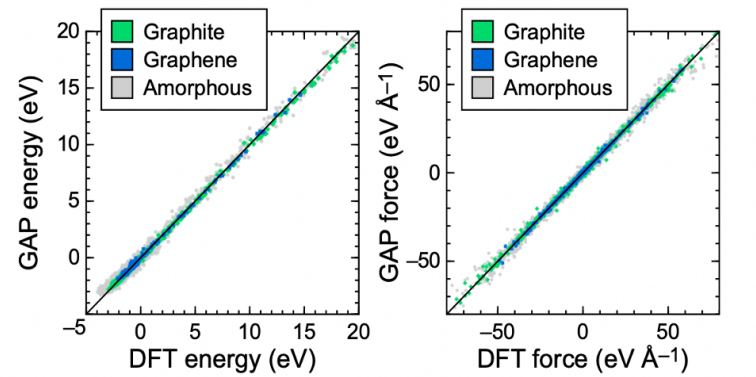

Figure 5: Plot of the energy and force predictions of a GAP potential compared with the associated DFT energies and forces. The data is of a MD simulation of lithium adsorption in different carbon nano-structures [8]

can also reproduce dynamical quantities, like the conductivity [15] or the phonon dispersion relation [13]. In Figure 6 one can see a plot of a conductivity computation of NaCl at different temperatures. Impressive about these results is first their accuracy compared to the experiment but second, that these result come out of an MD simulation running on an ML potential that was simply fitted to DFT calculations. No more physics were included when deriving this observable. Getting theses results would have been possible with a classical force field method too, but the amount of effort that has got to be put in constructing that force field makes the ML approach superior.

5 Limitations

One of the biggest obstacles of using machine learning potentials is the amount of data that is needed to obtain a sufficiently accurate energy surfaces and since in the end they are only a fit to the results of a electronic structure calculation, they first rely on these methods and second can, by definition, not lead to better results. While their purely mathematical form allows a highly flexible fit and is not tied to certain bonding parameters like classical force fields, this also impedes the interpretation of the model and can sometimes lead to "nonphysical" behavior when the ML potential is not trained sufficiently on a certain configuration [4]. Extrapolation is also a big aspect as the reliability of predictions for unknown configurations decreases drastically [11]. A last and highly debated topic is the assumption that the total energy of the system is equal to the sums of the atomic energies. There are novel approaches to include electrostatics but it is controversial if that is sufficient [9].

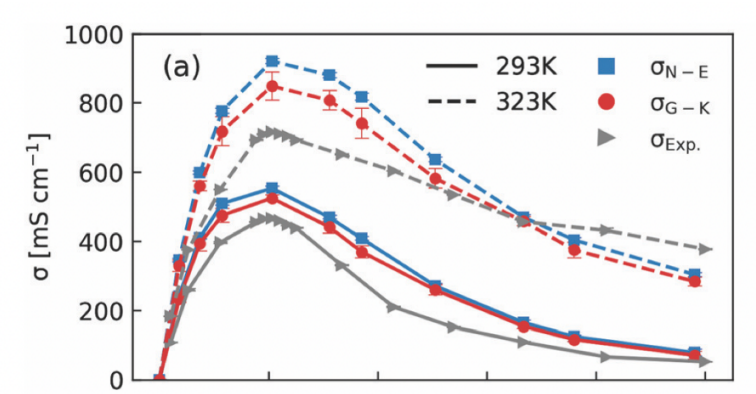

Figure 6: Conductivity computation using different theoretical methods. The underlying MD simulations were carry out using a ML potential. The blue data represent the evaluation with the Nernst Einstein method, the red data the Green Kubo approach. The gray data are the experimental observations [15].

6 Outlook

A forth generation of NNPs promise to include also long range interaction and charge transfer [9]. This is done by using a second high dimensional neuronal network to compute the charges in the system that are then used to first calculate an electronic energy and second also feed the charge distribution of the system into the atomic neuronal networks of the short distance energy evaluation. This charge transfer is especially important for long molecules and their dynamics as forces on atoms in molecules are responsible of the deformation and bending of the molecule. Another novel approach to make sure that the training data for neuronal networks contains a representative sample of the configuration space and that there are no vital configuration missing, is the so called *active learning*. One trains several neuronal networks with slightly different architectures on the same training set. Some of the NNP are then used to perform MD simulations with. The configurations contained in the resulting trajectory are then passed to one of the remaining neuronal networks to evaluate the energy at each point. If the energy values do differ for the creating NNP and the evaluating NNP, one has a clear idea of which configurations are not represented well enough in the training data [5].
Although the field of machine learned potentials is quite new, it is clear that future MD simulations will rely more and more on such trained potentials and more research has to be done, to improve sampling efficiency and also inclusion of physical phenomena. A major improvement would be to reduce the number of unique configurations to an amount that allows the usage of coupled cluster analysis. As MD simulations will become even more important in the future also the description of the energy surface will become increasingly important. Machine learned potentials are one approach to close the gap between highly accurate but slow electronic structure calculations on the one hand and fast but less accurate classical force fields on the other.

References

[1] Albert P. Bartók and Gabor Csanyi. “Gaussian approximation potentials: A brief tutorial introduction”. In: *International Journal of Quantum Chemestry* (2015). DOI: `https://doi.org/10.1002/qua.24927`.

[2] Albert P. Bartók, Risi Kondor, and Gábor Csányi. "On representing chemical environments". In: *Physical Review B* 87.18 (May 2013). ISSN: 1550-235X. DOI: 10.1103/physrevb.87.184115. URL: http://dx.doi.org/10.1103/PhysRevB.87.184115.

[3] Albert P. Bartók, Risi Kondor, and Gábor Csányi. "On representing chemical environments". In: *Phys. Rev. B* 87 (18 May 2013), p. 184115. DOI: 10.1103/PhysRevB.87.184115. URL: https://link.aps.org/doi/10.1103/PhysRevB.87.184115.

[4] J. Behler. "Constructing high-dimensional neural network potentials: A tutorial review". In: *International Journal of Quantum Chemistry* (2015). DOI: https://doi.org/10.1002/qua.24890.

[5] Jörg Behler. "Four Generations of High-Dimensional Neural Network Potentials". In: *Chemical Reviews* 0.0 (0). PMID: 33779150, null. DOI: 10.1021/acs.chemrev.0c00868.

[6] Jaime G. Carbonell, Ryszard S. Michalski, and Tom M. Mitchell. "AN OVERVIEW OF MACHINE LEARNING". In: *Machine Learning*. Ed. by Ryszard S. Michalski, Jaime G. Carbonell, and Tom M. Mitchell. San Francisco (CA): Morgan Kaufmann, 1983, pp. 3–23. ISBN: 978-0-08-051054-5. DOI: https://doi.org/10.1016/B978-0-08-051054-5.50005-4. URL: https://www.sciencedirect.com/science/article/pii/B9780080510545500054.

[7] J.and et. al. Finkbeiner. "Efficient Data Selection Methods for the Development of Machine Learning Potentials". In: *ICLR 2021 SimDL Workshop* (2021). DOI: notpublishedyet.

[8] So Fujikake et al. "Gaussian approximation potential modeling of lithium intercalation in carbon nanostructures". In: *The Journal of Chemical Physics* 148.24 (2018), p. 241714. DOI: 10.1063/1.5016317.

[9] T.W. Ko, J.A. Finkler, and S. et al Goedecker. "A fourth-generation high-dimensional neural network potential with accurate electrostatics including non-local charge transfer". In: *Nat Commun* 12 (398 2021). DOI: https://doi.org/10.1038/s41467-020-20427-2.

[10] Marcel F. Langer, Alex Goeßmann, and Matthias Rupp. *Representations of molecules and materials for interpolation of quantum-mechanical simulations via machine learning*. 2021. arXiv: 2003.12081 [physics.comp-ph].

[11] C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006. ISBN: 026218253X.

[12] M. Rupp. "Machine Learning for Quantum Mechanics in a Nutshell". In: *International Journal of Quantum Chemistry* (2015). DOI: https://doi.org/10.1002/qua.24954.

[13] Y. Shaidu and R. et al Küçükbenli E.and Lot. "A systematic approach to generating accurate neural network potentials: the case of carbon". In: *npj Comput Mater* 7 (52 2021). DOI: https://doi.org/10.1038/s41524-021-00508-6.

[14] Alain B. Tchagang and Julio J. Valdés. "Prediction of the Atomization Energy of Molecules Using Coulomb Matrix and Atomic Composition in a Bayesian Regularized Neural Networks". In: *Lecture Notes in Computer Science* (2019), pp. 793–803. ISSN: 1611-3349. DOI: 10.1007/978-3-030-30493-5_75. URL: http://dx.doi.org/10.1007/978-3-030-30493-5_75.

[15] S. Yunqi, M. Hellström, and J. et al Behler. "Temperature effects on the ionic conductivity in concentrated alkaline electrolyte solutions". In: *Phys. Chem. Chem. Phys.* 22 (2020), pp. 10426–10430. DOI: DOI:10.1039/C9CP06479F.