PAPER • OPEN ACCESS

On the role of gradients for machine learning of molecular energies and forces

View the article online for updates and enhancements.

MACHINE
LEARNING
Science and Technology

PAPER

On the role of gradients for machine learning of molecular energies and forces

Anders S Christensen and O Anatole von Lilienfeld

Institute of Physical Chemistry National Center for Computational Design and Discovery of Novel Materials (MARVEL), Department of Chemistry, University of Basel, Klingelbergstrasse 80, CH-4056 Basel, Switzerland

E-mail: anatole.vonlilienfeld@unibas.ch

Abstract

The accuracy of any machine learning potential can only be as good as the data used in the fitting process. The most efficient model therefore selects the training data that will yield the highest accuracy compared to the cost of obtaining the training data. We investigate the convergence of prediction errors of quantum machine learning models for organic molecules trained on energy and force labels, two common data types in molecular simulations. When training models for the potential energy surface of a single molecule, we find that the inclusion of atomic forces in the training data increases the accuracy of the predicted energies and forces 7-fold, compared to models trained on energy only. Surprisingly, for models trained on sets of organic molecules of varying size and composition in non-equilibrium conformations, inclusion of forces in the training does not improve the predicted energies of unseen molecules in new conformations. Predicted forces, however, improve about 7-fold. For the systems studied, we find that force labels and energy labels contribute equally *per* label to the convergence of the prediction errors. The optimal choice of what type of training data to include depends on several factors: the computational cost of acquiring the force and energy labels for training, the application domain, the property of interest and the complexity of the machine learning model. Based on our observations we describe key considerations for the creation of new datasets for potential energy surfaces of molecules which maximize the efficiency of the resulting machine learning models.

1. Introduction

In recent years, machine learning models have become increasingly popular as methods to approximate potential energy surfaces of molecules. These models range from classical learning methods such as kernel methods to methods based on deep neural networks [1–19]. A common denominator for these data-driven models is that they require an adequate training set in order to yield predictions of sufficient accuracy. It is thus clear that informed and rational selection of training data is paramount to proper optimization of the data-efficiency of the machine learning models.

For machine learning models describing potential energy surfaces, two types of data seem particularly convenient as training data: single-point energies and atomic force vectors. However, it has not yet been fully demonstrated when and to which degree the inclusion of force labels in the training set truly leads to an improvement of the accuracy of the trained model versus energy labels. It is even possible to find somewhat conflicting information in literature. For example, the GDML and sGDML methods achieve state-of-the-art accuracy in certain cases for the MD17 benchmark dataset, despite training only on force labels, ignoring the energy labels [20–22]. In contrast, the HIP-NN neural network—when only trained on energy labels—ostensibly achieves similar predictive accuracy to the DTNN, SchNet and PhysNet neural networks, when these are trained on both forces *and* energy labels for 50 K molecules from the MD17 dataset, despite the fact that HIP-NN is trained using far fewer training labels [15, 19, 23, 24]. Similarly, the family of ANI datasets have resulted in successful potential energy models for general organic chemistry solely trained on

single-point energies on geometries distorted along normal-modes, although the corresponding gradient information would not have been much more costly to obtain at the density-functional theory (DFT) level employed [5, 25–27].

Consequently, it is not clear from literature which is the best strategy for designing a dataset for the most accurate machine learning potentials for chemical compounds within a given budget of computational resources. In order to shed more light on this matter, here we investigate how the predictive accuracy of machine learning models is affected by inclusion of derivative information—namely atomic force vectors in addition to the energy—in the training set. More specifically, we investigate the nine possible combinations resulting from the inclusion of functions, derivatives and functions plus derivatives in loss-functions used for training, as well as in error measures used for evaluating the prediction error.

In order to *learn*, the predictive accuracy of the trained machine *must* strictly increase with an increasing amount of training data, aside from statistical variation. This holds true as long as the training data is noiseless and the machine learning model is flexible enough to be able to properly account for all variation in the data [28, 29]. This has previously been demonstrated numerically with the leading term of the prediction error decreasing according to a power-law with number of training samples, N, for kernel ridge regression [30], as well as for neural networks [31], i.e.:

$$\text{Error} \propto \frac{a}{N^b} \tag{1}$$

Learning curves of functional machine learning models thus *must* behave linearly on log-log scales and form, in the limit of large N, a robust tool to assess learning data-efficiency through comparison of the 'offset,' $\log(a)$ and 'slope,' b.

This paper is structured as follows: we first describe our methodology and illustrate the idea of learning curve comparison for a well-known toy system from mathematics, the well-described Himmelblau's function, in order to demonstrate how different types of training data affect the learning rate for a generic non-trivial 2-dimensional surface [32]. Next, we employ the same testing scheme for two different but common use cases in chemistry: The first use case concerns generating a force field for a single molecule. Here we demonstrate how using forces and energies in the training data affect the learning rate for the potential energy surfaces of 10 individual molecules from the MD17 dataset [20, 21]. In order to ensure that the underlying data is practically noise-free, we additionally present a revision of the MD17 where the energies and forces have been calculated using dense integration grids and tight self-consistent field (SCF) tolerances.

The second use case concerns generating machine learning models trained *across* chemical compound space, meaning that they can extrapolate to out-of-sample molecules not seen during training. Also here, we investigate how the inclusion of forces in the training data helps the trained machine to generalize from points on the potential energy surfaces of known molecules to points on the potential energy surfaces of new molecules.

2. Theory

In order to achieve a fair comparison between models trained on different kinds of data (here energies and forces), we employ three closely related kernel-based regression models which are described in this section, as well as standard training protocols [33]. Using these schemes, we ensure that models trained on different types of data will rely on the same kernel functions and representations while at the same time being exactly-determined with closed-form solutions. This is done by defining loss functions in which basis kernel functions are placed on the training data, as well as placing kernel derivatives corresponding to any gradient or force information in the training data. This yields a set of basis functions which guarantees that the model (within a given regularization) is able to perfectly align with the training labels. Additionally, use of the force operator ensures that the resulting force fields are conservative force fields, which is important for molecular dynamics simulations and that the predicted energies obey common physical relations such as rotational and translational invariances.

The three machine learning models which employ training on either energies (function) only, or energies and forces (function and derivatives) simultaneously, or forces (derivatives) only are presented next. Each model is characterized by its loss function J which is minimized through regression. Note, that prediction errors can also be evaluated using different loss functions. Conventionally, training and test error definition are identical, and their evaluation only differs in being assessed on a training or test set, respectively. As such, there are nine possibilities to combine the three loss functions in training and in testing.

While the relevant equations for each regressor can be derived both in the context of kernel ridge regression (KRR), as well as Gaussian process regression, we present them here in the notation most

commonly used for the former [10, 14, 20–22, 34–38]. For more detailed derivations, we refer to the work by Bartók and Csányi [1], as well as that of Mathias [39]. We further note that the equations for the force-only regressor have recently seen a different derivation in the work on 'gradient domain' machine learning (GDML) by Chmiela hbox*et al* [20].

2.1. Energy-only training

In KRR, the energy, u, of a query molecule, represented by $\tilde{M}$, can be expanded using a basis set of kernel functions placed on training samples. That is,

$$u\left(\tilde{M}\right) = \sum_{i}^{\text{training}} \|\left(\tilde{M}, M_i\right) \alpha_i, \tag{2}$$

where M_i is the ith molecule in the training set, α_i is the ith regression coefficient with units of energy and $\|(\cdot, \cdot)$ is a function that relates two molecules through a similarity measure which typically depends on the specific functional form of the kernel function, the representation and the metric. Writing the above equation on matrix form yields:

$$\mathbf{u} = \mathbf{K}\alpha \tag{3}$$

Here, $\mathbf{u}$ is the vector containing predicted energies, $\mathbf{K}$ is the kernel matrix containing the pairwise kernel elements between the molecules in the training set and the predictions sets and finally α is the vector containing regression coefficients. Bold lowercase letters indicate vectors and bold capital letters indicate matrices. The specific kernel function used for the different experiments are given in section 2.4. From equation (3), a set of regression coefficients can be obtained by minimizing a loss function. Using Tikhonov regularization and minimizing the squared errors leads to the loss function which is the foundation of KRR:

$$J(\alpha \mid \{\mathbf{u}^{\text{ref}}\}) = \frac{1}{2}\|\mathbf{K}\alpha - \mathbf{u}^{\text{ref}}\|_2^2 + \frac{\lambda}{2}\alpha^\top \mathbf{K}\alpha \tag{4}$$

This loss function contains a hyperparameter, λ, which determines the amount of L2-regularization for the regression coefficients stored in α, which in turn yields the well-known closed-form solution for training a KRR model:

$$\alpha = (\mathbf{K} + \mathbf{I}\lambda)^{-1} \mathbf{u}^{\text{ref}}. \tag{5}$$

These regression coefficients can then be used to predict energies (as in equation (3)), through evaluation of the relevant kernel matrix. Via the definition of the force operator, i.e. the negative gradient of the energy with respect to the atomic coordinate vector, $\mathbf{r}$, the atomic force vector, $\mathbf{f}$, can be predicted by differentiation of equation (3):

$$\mathbf{f} \triangleq -\frac{\partial}{\partial \mathbf{r}}\mathbf{u} = \left[-\frac{\partial}{\partial \mathbf{r}}\mathbf{K}\right] \alpha \tag{6}$$

Note that this approach is as 'naive' as general and in principle any differential property can be learned this way as long as the representation accounts for all the variables which are being perturbed [38].

2.2. Combined energy and force training

The second model investigated herein, is a model closely related to conventional KRR but which allows for training on both force and energies simultaneously. Similar to how KRR can be used to construct a function that goes through the training points exactly, it is also possible to—at the same time—enforce the derivatives at those points. Here, derivatives are enforced by including the derivatives (for example, forces) in the regression in addition to the function value (for example, energies). In order to allow the model to match both the function values and derivatives of the training set exactly, it is necessary to increase the number of basis functions considerably, as the set of equations would otherwise be vastly overdetermined, as, for example, there are 3 N force labels for each energy label. One choice of extended basis comes from augmenting the set of basis functions in standard KRR with additional kernel functions corresponding to the kernel derivatives with respect to the coordinates of the training molecules. This choice of basis ensures that the basis function accompanying a training label always adds the necessary flexibility to describe that label exactly. The resulting set of equations for this problem looks as follows:

$$\begin{bmatrix} \mathbf{u} \\ \mathbf{f} \end{bmatrix} = \begin{bmatrix} \mathbf{K} & -\frac{\partial}{\partial \mathbf{r}^\top}\mathbf{K} \\ -\frac{\partial}{\partial \mathbf{r}}\mathbf{K} & \frac{\partial^2}{\partial \mathbf{r}\partial \mathbf{r}^\top}\mathbf{K} \end{bmatrix} \alpha \tag{7}$$

Here, $\mathbf{K}$ is again the kernel matrix containing the pairwise kernel elements between the molecules in the training set and the predictions sets and is defined identically to $\mathbf{K}$ in equation (3). Similarly to the problem in equation (3), this choice of basis functions ensures that the training kernel is square-symmetric and the minimization of the regularized loss function has a convenient closed-form solution. For a set of reference training energies and forces, the regression coefficients can here be obtained by minimizing the following set of squared errors with Tikhonov regularization:

$$J(\boldsymbol{\alpha} \mid \{\mathbf{u}^{\text{ref}}, \mathbf{f}^{\text{ref}}\}) = \frac{1}{2} \left\| \begin{bmatrix} \mathbf{K} & -\frac{\partial}{\partial \mathbf{r}^{\intercal}}\mathbf{K} \\ -\frac{\partial}{\partial \mathbf{r}}\mathbf{K} & \frac{\partial^2}{\partial \mathbf{r}\partial \mathbf{r}^{\intercal}}\mathbf{K} \end{bmatrix} \boldsymbol{\alpha} - \begin{bmatrix} \mathbf{u}^{\text{ref}} \\ \mathbf{f}^{\text{ref}} \end{bmatrix} \right\|_2^2 + \frac{\lambda}{2}\boldsymbol{\alpha}^{\top} \begin{bmatrix} \mathbf{K} & -\frac{\partial}{\partial \mathbf{r}^{\intercal}}\mathbf{K} \\ -\frac{\partial}{\partial \mathbf{r}}\mathbf{K} & \frac{\partial^2}{\partial \mathbf{r}\partial \mathbf{r}^{\intercal}}\mathbf{K} \end{bmatrix} \boldsymbol{\alpha} \qquad (8)$$

The closed-form solution is, similarly to that of equation (5),

$$\boldsymbol{\alpha} = \left(\begin{bmatrix} \mathbf{K} & -\frac{\partial}{\partial \mathbf{r}^{\intercal}}\mathbf{K} \\ -\frac{\partial}{\partial \mathbf{r}}\mathbf{K} & \frac{\partial^2}{\partial \mathbf{r}\partial \mathbf{r}^{\intercal}}\mathbf{K} \end{bmatrix} + \mathbf{I}\lambda \right)^{-1} \begin{bmatrix} \mathbf{u}^{\text{ref}} \\ \mathbf{f}^{\text{ref}} \end{bmatrix}. \qquad (9)$$

This definition guarantees that the regression problem is exactly determined and allows for the machine-learned potential energy surface to match both the energy and the forces of each training molecule.

With the regression coefficients obtained through equation (9), predicted energies can then be calculated via the set of kernel functions and kernel derivatives placed on the training set:

$$\mathbf{u} = \begin{bmatrix} \mathbf{K} & -\frac{\partial}{\partial \mathbf{r}^{\intercal}}\mathbf{K} \end{bmatrix} \boldsymbol{\alpha} \qquad (10)$$

Likewise, forces can be obtained by taking the relevant derivative of the energy, i.e.:

$$\mathbf{f} \triangleq -\frac{\partial}{\partial \mathbf{r}}\mathbf{u} = \begin{bmatrix} -\frac{\partial}{\partial \mathbf{r}}\mathbf{K} & \frac{\partial^2}{\partial \mathbf{r}\partial \mathbf{r}^{\intercal}}\mathbf{K} \end{bmatrix} \boldsymbol{\alpha} \qquad (11)$$

2.3. Force-only training

The third model presented is a model which constructs a potential energy surface from information about derivatives only. This model is similar to the model described in section 2.2 except that, in this case, only force labels are used in the training step and, consequently, the set of basis functions is comprised of only the corresponding kernel derivatives.

In this approach, we start with the following equation, in which the kernel matrix is formed by the double derivatives of pair-wise kernel functions:

$$\mathbf{f} = \begin{bmatrix} \frac{\partial^2}{\partial \mathbf{r}\partial \mathbf{r}^{\intercal}}\mathbf{K} \end{bmatrix} \boldsymbol{\alpha} \qquad (12)$$

For a set of reference force labels, $\mathbf{f}^{\text{ref}}$, the corresponding regression coefficients are obtained analogously to the energy-only and energy+force examples in the previous sections.

$$J(\boldsymbol{\alpha} \mid \{\mathbf{f}^{\text{ref}}\}) = \frac{1}{2} \left\| \begin{bmatrix} \frac{\partial^2}{\partial \mathbf{r}\partial \mathbf{r}^{\intercal}}\mathbf{K} \end{bmatrix} \boldsymbol{\alpha} - \mathbf{f}^{\text{ref}} \right\|_2^2 + \frac{\lambda}{2}\boldsymbol{\alpha}^{\top} \begin{bmatrix} \frac{\partial^2}{\partial \mathbf{r}\partial \mathbf{r}^{\intercal}}\mathbf{K} \end{bmatrix} \boldsymbol{\alpha} \qquad (13)$$

$$\boldsymbol{\alpha} = \left(\begin{bmatrix} \frac{\partial^2}{\partial \mathbf{r}\partial \mathbf{r}^{\intercal}}\mathbf{K} \end{bmatrix} + \mathbf{I}\lambda \right)^{-1} \mathbf{f}^{\text{ref}}. \qquad (14)$$

This is also sometimes referred to as training in the 'gradient domain' [20].

Energies can then be predicted through direct integration of equation (12), which allows the scalar field to be determined up to an integration constant:

$$\mathbf{u} = \begin{bmatrix} -\frac{\partial}{\partial \mathbf{r}^{\intercal}}\mathbf{K} \end{bmatrix} \boldsymbol{\alpha} + \text{constant term} \qquad (15)$$

For problems where only a single surface is of interest—for example the potential energy surface of a given stoichiometry—the integration constant is rarely of any importance and can also be inferred by predicting the energies for the training data. Unfortunately, however, this approach is less practical when multiple surfaces are of interest, for example for datasets involving the potential energy surfaces of multiple different molecules. Note that the number of possible stoichiometries is known to grow exponentially with elementary particle number [40]. Hence, force-only training makes it hard if not impossible to train models that directly predict energies for molecules of varying size and chemical composition, if the energy of the molecules are of interest. Of course, composite approaches, e.g. using dressed atom models [41], can still be used to rectify such shortcomings.

2.4. Representations and kernel functions

Since we present machine learning models trained on (i) an analytical function (Himmelblau's function) and on (ii) several models of molecular energetics trained non-equilibrium geometries, different kernel and representation choices have been made.

2.4.1. Representing Himmelblau's function

Here, Himmelblau's function is used as a toy-model for learning complicated surfaces [32]. This allows for thorough benchmarking of the regressors used herein. Himmelblau's function is a multi-modal function with one local maximum and four minima and bears some resemblance to the potential energy surface of a simple molecule with four conformational minimia. The function is defined as:

$$u(x,y) = \left(x^2 + y - 11\right)^2 + \left(x + y^2 - 7\right)^2. \tag{16}$$

Points on the surface are represented here as their xy-coordinate pair, that is, $\mathbf{q}_i = [x_i, y_i]^\top$. A Gaussian kernel function is used to compute the kernel elements for Himmelblau's function, i.e.:

$$\mathbf{K}_{ij} = \exp\left(-\frac{\|\mathbf{q}_i - \mathbf{q}_j\|_2^2}{2\sigma^2}\right) \quad \text{for Himmelblau's function} \tag{17}$$

This choice of representation and kernel function makes is straightforward to implement and enables the computation of necessary kernels and derivatives analytically.

2.4.2. Representing molecules

To represent the environments of an atom in a molecule, we rely on the computationally more efficient variant of the Faber–Christensen–Huang–Lilienfeld (FCHL) representation [42], namely the FCHL19 representation [22]. Briefly, this representation is a vector which contains histograms of the radial distributions of atoms and a number of Fourier terms describing angular distributions of atoms in the environment of a certain atom [43]. In principle, any continuous representation that generalizes across chemical space could have been used for this purpose. In addition, we use the localized kernel ansatz in which the kernel elements between two molecules correspond to the pairwise sum over the kernel functions between the respective representations of atomic environments in the two molecules [37, 44]. This makes it possible to train models that span molecules of varying size and chemical composition. The following Gaussian kernel function is used throughout:

$$\mathbf{K}_{ij} = \sum_{I \in i} \sum_{J \in i} \delta_{Z_I Z_J} \exp\left(-\frac{\|\mathbf{q}_I - \mathbf{q}_J\|_2^2}{2\sigma^2}\right) \quad \text{for molecules} \tag{18}$$

where $\mathbf{q}_I$ and $\mathbf{q}_J$ are the representations of the Ith and Jth atoms in the molecules i and j, respectively, δ is the Kronecker delta, with Z_I and Z_J being the atomic numbers of each atom, respectively.

3. Results

In this section, we present numerical evidence which demonstrates the effects on the learning rate of including derivative labels in the training data. This section is organized as follows: first, we establish the generality of our numerical experiment by learning the surface of a simple two-dimensional function, unrelated to molecules or chemistry. Next, we demonstrate the same principles applied to two distinct use cases, namely (1) training a model for the potential energy surface for a single molecule and (2) training a general model for the potential energy surfaces of a number of molecules of different size and chemical composition.

3.1. Toy system: learning Himmelblau's function

In this section we investigate the learning curves of machines trained to predict Himmelblau's Function. This 2D surface has four local minima and is displayed in figure 1, as well as the norm of its gradient. Here, Himmelblau's function serves as a toy system to demonstrate the effects of including function derivatives in the training procedure.

We generate a dataset by selecting random points from the surface. The test set consists of 10 000 points sampled randomly and uniformly across the interval $\{x_i, y_i\} \in [-6.0; 6.0]$, with training sets of varying sizes sampled randomly and uniformly across the same interval. For each training set size $N \in \{25, 50, 100, 200, 400, 800, 1600\}$, 100 training and test sets are created using 100-fold random sub-sampling cross-validation with a constant test set size of 10 000 points. This extensive cross-validation
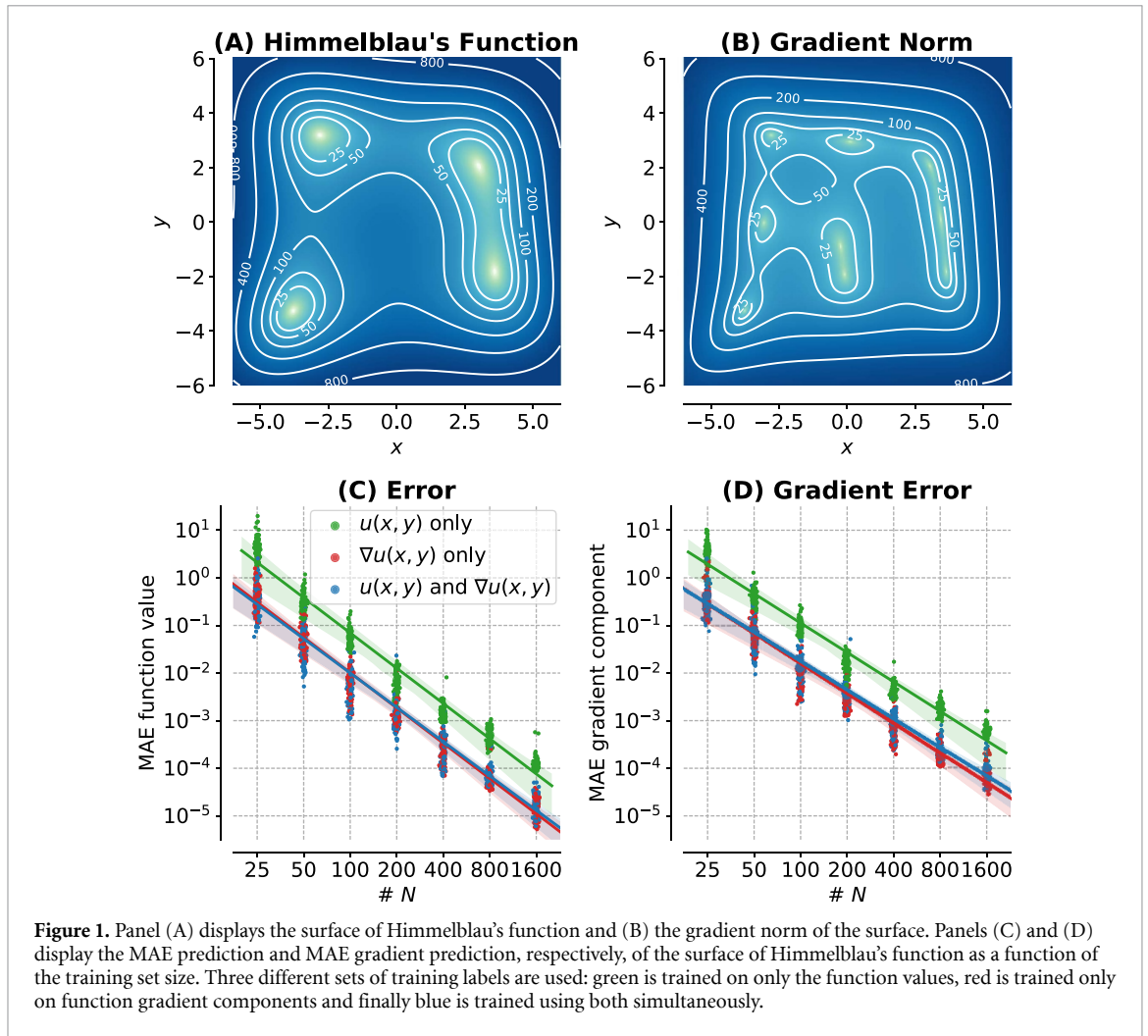
Figure 1. Panel (A) displays the surface of Himmelblau's function and (B) the gradient norm of the surface. Panels (C) and (D) display the MAE prediction and MAE gradient prediction, respectively, of the surface of Himmelblau's function as a function of the training set size. Three different sets of training labels are used: green is trained on only the function values, red is trained only on function gradient components and finally blue is trained using both simultaneously.

Table 1. Slopes and offsets for predictions of the MAE value and MAE gradient component for Himmelblau's function, based on three machine learning methods trained on different types of data.

Training data	MAE value		MAE gradient	
	Slope	Offset	Slope	Offset
Values	-2.45 ± 0.03	8.5 ± 0.2	-2.05 ± 0.03	7.1 ± 0.2
Values and gradients	-2.33 ± 0.03	5.8 ± 0.2	-1.96 ± 0.03	4.8 ± 0.2
Gradients	-2.43 ± 0.03	6.4 ± 0.2	-2.06 ± 0.03	5.2 ± 0.2

was necessary in order to have well-converged averages over the folds, as the test errors were observed to vary up to one order of magnitude. In total over 350 000 individual models with different training sets and hyperparameters were trained in order to obtain the presented learning curves.

Three different models were trained for the surface: the first model trained on only the function values, the second only trained on the function derivatives at the training points and the third model was trained on both the function values and function derivatives simultaneously. As expected [30, 31], the three machines yield learning curve fits for the predicted mean-absolute-error (MAE) function value with similar slopes when plotted on a log-log scale, as displayed in figure 1(C).

The average difference between the two models which include forces labels in the training set is much smaller than the standard deviation of the 100 folds used to calculate the average. Compared to training on function values exclusively, the average decrease in the off-set of the learning-curve was found to be a factor of 7.0 for the model trained on only function derivatives and 7.2 for the model trained on both function values and function derivatives.

Learning curves for gradient predictions show similar trends among the three models (see figure 1(D): the average decrease in the MAE of function values predicted on a test set was found to be a factor of 7.9,

Table 2. Learning curves for three machines trained on different types of data and the same loss functions evaluated on a test set of 10 000 points on the surface of Himmelblau's function. The three machines are trained on either function values, function values and gradients simultaneously, or gradients only. The loss functions used to train each machine is denoted in the left column, followed by the slopes and offsets for the resulting learning curves for three loss functions evaluated on the test data. The three loss functions are defined in the text in equation (4), 8 and 13, respectively.

Training		Test loss $J(\boldsymbol{\alpha} \mid \{\mathbf{u}^{\text{test}}\})$		Test loss $J(\boldsymbol{\alpha} \mid \{\mathbf{u}^{\text{test}}, \mathbf{f}^{\text{test}}\})$		Test loss $J(\boldsymbol{\alpha} \mid \{\mathbf{f}^{\text{test}}\})$	
Training data	Training loss	Slope	Offset	Slope	Offset	Slope	Offset
Values	$J(\boldsymbol{\alpha} \mid \{\mathbf{u}^{\text{ref}}\})$	-4.59 ± 0.03	27.1 ± 0.2	-3.74 ± 0.03	25.2 ± 0.2	-3.63 ± 0.03	24.4 ± 0.2
Values and gradients	$J(\boldsymbol{\alpha} \mid \{\mathbf{u}^{\text{ref}}, \mathbf{f}^{\text{ref}}\})$	-4.41 ± 0.03	22.4 ± 0.2	-3.58 ± 0.03	20.8 ± 0.2	-3.49 ± 0.03	20.3 ± 0.2
Gradients	$J(\boldsymbol{\alpha} \mid \{\mathbf{f}^{\text{ref}}\})$	-4.60 ± 0.03	23.1 ± 0.2	-3.78 ± 0.03	21.5 ± 0.2	-3.69 ± 0.03	20.9 ± 0.2

when training on gradients only and 7.0 when training on both function values and gradients simultaneously, compared to training only on function values.

Fitted slopes and offsets for all six learning curves are displayed in table 1. We note that the 95% confidence intervals of the fitted learning curves for the two models that include gradients in the training loss-function are mostly overlapping and while slopes and offsets differ somewhat (see table 1), the difference between the two models is not statistically significant.

Also for learning curves using loss functions we find agreement with the power-law behavior that is expected from models trained on function values [30, 31], here demonstrated for models trained on function gradients. In table 2, we present resulting slopes and offsets for loss-function learning curves for the three types of trained machines trained using three different sets of training data and corresponding loss functions (rows). These learning curves are shown graphically in figure S1 in the Supplementary Information (stacks.iop.org/MLST/1/045018/mmedia).

Results for the corresponding three test loss functions (columns) are obtained using a test set of 10 000 points randomly selected from the surface of Himmelblau's function. Here, the same trends as for the MAE learning curves are observed. The resulting slopes from the three types of trained machines do not differ with statistical significance, but instead depend on the type of test data. In contrast, the offsets depend strongly on the training loss function and training data. More specifically, the inclusion of gradients in the training loss function is on average beneficial when predicting function values, function and gradient values, as well as gradient values alone. We also remind the reader that the slope of these learning curves are independent of the units of the labels, as this quantity is folded into the offset. Despite using hundred-fold cross-validation, we do not observe a statistically significant difference between any of the three test predictions when using training loss functions and data sets based on function values and gradients, or gradients alone. We also note that the learning curves for the MAE error are observed to follow the expected power-law behavior over a range that spans 6 and 5 orders of magnitudes for predicted function values and predicted function gradient components, respectively.

We have thus demonstrated for the Himmelblau function, that MAE error in this case can be decreased 7-fold by inclusion of labels that correspond to function derivatives in the training set. At the same time, however, we find negligible improvement upon inclusion of function values, in addition to the gradient information in the training algorithm. Interestingly, these results suggest that when learning a single function surface, using derivatives as training labels is more advantageous than actual function values.

3.2. Use case 1: learning the PES for a one molecule

In this section we discuss the accuracy of energy and force predictions from three models trained on the revised MD17 dataset [20, 21].

For each of the 10 molecules in the revised MD17 dataset, three models are trained using either only the energy labels of each sample, only the force labels, or both types of labels simultaneously. For each of the three models, the training set size N is varied for $N \in \{50, 100, 150, 200, 400, 600, 800, 1000\}$. The resulting MAE of energy and force predictions for each molecule can be found in table S1 in the Supplementary Information and plots with linear fits to the learning curves are shown in figure 2. As it is clear from the results and similarly to the example with Himmelblau's function, including the forces in the training procedure improves the prediction of both energies and forces. On average, the MAE of the predicted energies and force components is reduced about 7-fold for the same number of training samples. Regarding the inclusion of energies in addition to forces, we find, again in complete analogy to the Himmelblau function, that it makes little difference. In all cases, the largest difference was found to be 0.001 kcal mol^{-1} MAE predicted energy and 0.002 kcal mol^{-1} Å^{-1} MAE predicted force components at the largest training set size of $N = 1000$ samples.
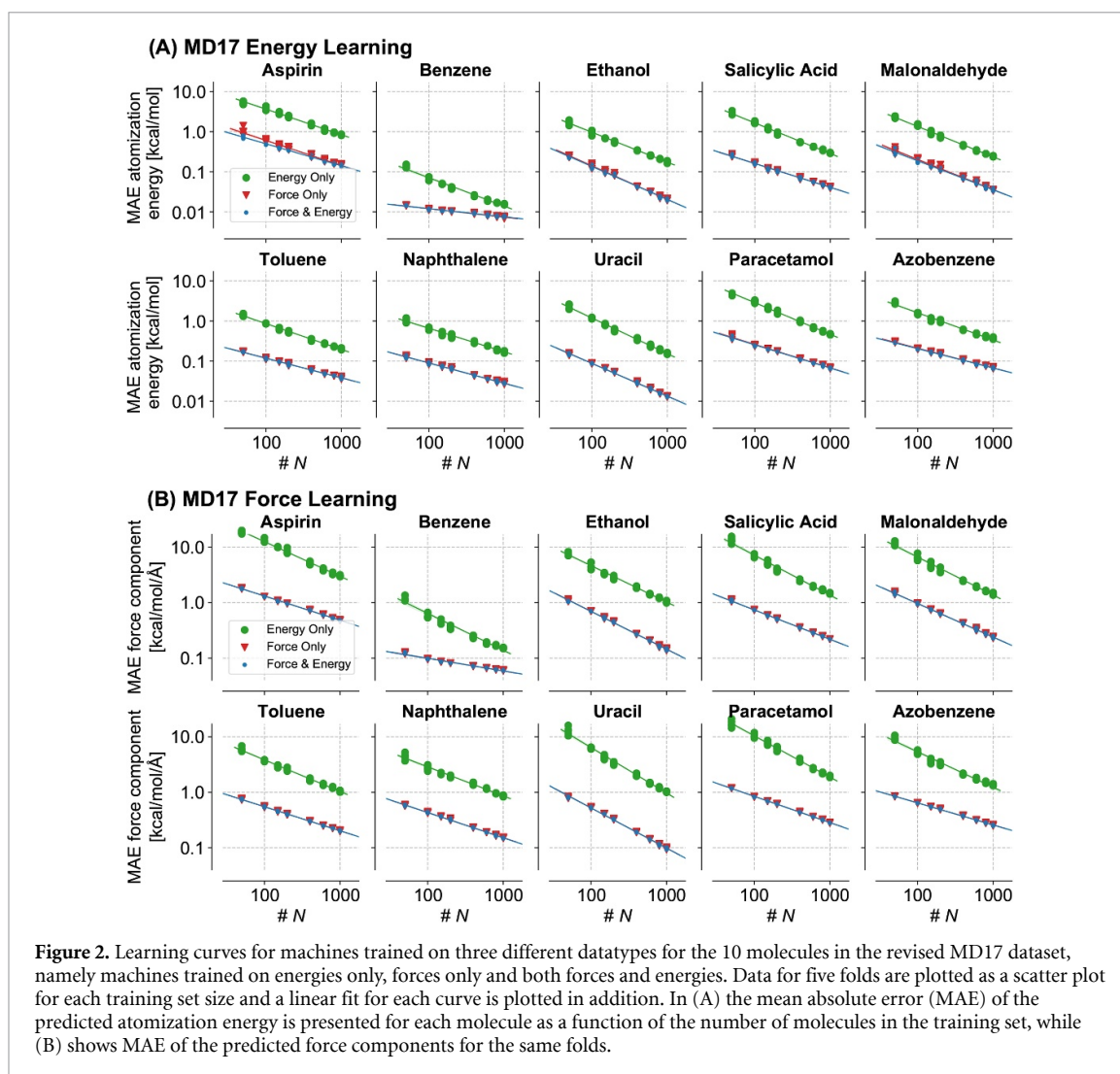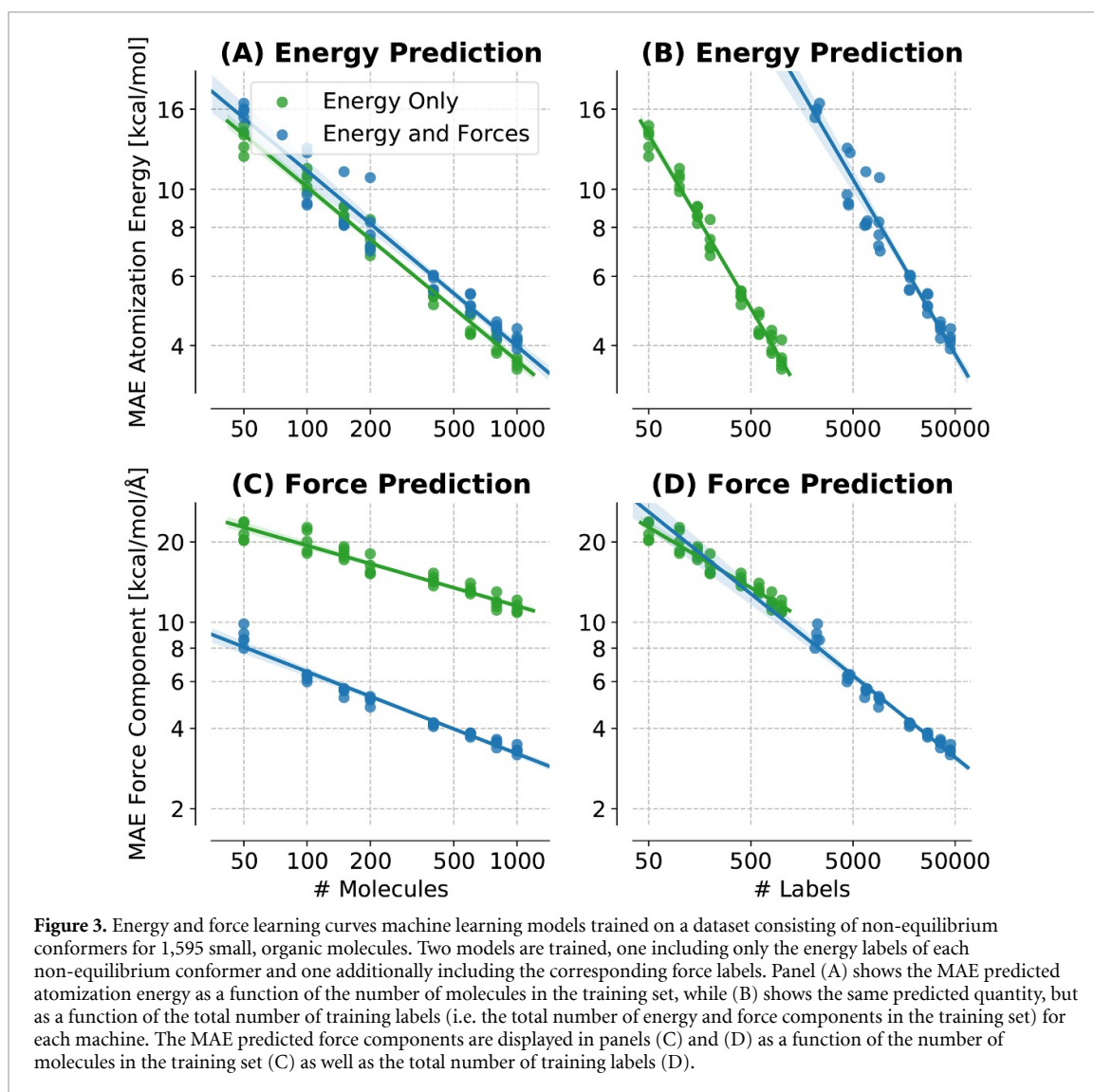
Figure 2. Learning curves for machines trained on three different datatypes for the 10 molecules in the revised MD17 dataset, namely machines trained on energies only, forces only and both forces and energies. Data for five folds are plotted as a scatter plot for each training set size and a linear fit for each curve is plotted in addition. In (A) the mean absolute error (MAE) of the predicted atomization energy is presented for each molecule as a function of the number of molecules in the training set, while (B) shows MAE of the predicted force components for the same folds.

In figure S2 the MAE predicted energy is plotted as a function of the number of training labels for each of the 10 molecules using the three models described earlier. Here, the slopes and offsets of the learning curve are close to identical in all 10 cases, regardless of what data was used to train the models. This suggests that for this dataset, one force label yields as much improvement in the predictive accuracy of a machine learning model as one energy label. Thus, a model for a specific molecule, trained on a certain number of energies, will have roughly the same predictive accuracy as one that is trained on the same number of force labels. As a consequence, in such cases—and if force-evaluations represent computationally negligible overhead (common within, for example, DFT)—the number of independent quantum calculations necessary to generate the training data required to reach a certain predictive power, can be reduced by a factor of $3N$ thanks to the inclusion of forces.

We find that in all but one case, the learning curves for both force as well as energy prediction follows the expected power law, meaning they display a linear relationship on a log-log scale. In the case of benzene, the off-set of the learning curve is extremely low, but the slope is inconsistent with the expected power-law. This can be seen clearly in figure S1 in the Supplementary Material. It seems unlikely that underlying numerical noise in the training data is at the origin of such premature saturation: the learning curves for Uracil are equally low and do not suffer from lack of linearity. The more likely culprit appears to be the specific parameterization of the FCHL19 representation achieving only insufficient uniqueness, a possibility recently pointed out by Pozdnyakov hbox*et al* [45]. The addition of higher order terms, such as for example 4-body terms, into FCHL [42] might rectify this issue. Furthermore, we note that the parameters in the representation have not been re-optimized for discriminating the subtle difference between the very similar structures in this, but rather work across molecules of varying sizes and chemical composition. It is thus possible that they could be re-optimized to alleviate such issues to some extent.

Figure 3. Energy and force learning curves machine learning models trained on a dataset consisting of non-equilibrium conformers for 1,595 small, organic molecules. Two models are trained, one including only the energy labels of each non-equilibrium conformer and one additionally including the corresponding force labels. Panel (A) shows the MAE predicted atomization energy as a function of the number of molecules in the training set, while (B) shows the same predicted quantity, but as a function of the total number of training labels (i.e. the total number of energy and force components in the training set) for each machine. The MAE predicted force components are displayed in panels (C) and (D) as a function of the number of molecules in the training set (C) as well as the total number of training labels (D).

Table 3. Slope and offsets (see text) of linear fits to mean absolute error (MAE) of energies and gradients, predicted on a test set, versus the training set size, on a log-log scale. The dataset consists of 1595 small organic molecules in non-equilibrium conformation (see text). The column 'Training' denotes what data was used to train the models. Lastly, the column 'N unit' denotes the unit is used for the abscissa of a given learning curve. [Molecules] denotes that the unit is simply the number of molecules in the training set, that is for energy-only training, one training label (the energy) is used per molecule, while for energy+forces, $3N+1$ training labels (one energy and $3N$ force components) are used for each molecule, whereas [Labels] denotes that the total number of $3N+1$ labels are used on the abscissa.

Training	N unit	MAE energy [kcal/mol]		MAE force [kcal/mol/Å]	
		Slope	Offset	Slope	Offset
Energy only	[Molecules]	−0.44	4.35	−0.23	4.01
Energy and forces	[Molecules]	−0.45	4.47	−0.31	3.29
Energy and forces	[Labels]	−0.45	6.16	−0.31	4.46

3.3. Use case 2: training models across chemical composition

In our second use case scenario, we investigate how learning forces and energy across chemical compound space behaves as a function of the number of training molecules and the type of training data. Figure 3 shows learning curves for force and energy predictions on a dataset consisting of non-equilibrium conformations of 1,595 small, organic molecules. The training set is divided such that every (distorted) molecule of a given chemical composition is only seen once in either the training or test sets. Using this data, two types of models are trained: one model is trained using only atomization energy labels and a second model is trained using both atomization energy as well as the corresponding $3N$ components of the force vector. As the model trained on only forces would only integrate the energy up to an arbitrary constant, the predicted energies would not be meaningful and therefore this model is not used in this section.

Figure 3(A) shows the MAE predicted atomization energy for the two models as a function of the number of molecules involved. The learning rates hardly differ, with the largest deviation between the two curves being 3% of the average MAE. While this seems surprising in light of the effect of including forces for the MD17 dataset discussed in the previous section, this result has a simple explanation. Since the gradient information only provides information about the underlying function up to an integration constant, the additional information is useless in providing information about the differences between the individual potential energy surface on which all the molecules are located. In this case, this information is provided solely by the atomization energies and, in the end, the dominant error in the learned atomization energies comes from learning energy differences between the different constitutions of atoms in each molecule and not from more subtle changes in molecular conformation. The learning rates are displayed in table 3, where the slopes and offsets for energy predictions are very close for the two different models. For energy predictions, the slopes of the learning curves are -0.44 and -0.45 for the models trained on only energies and the models simultaneously trained on energies and forces, respectively. For force prediction, however, the slopes differ somewhat, at -0.23 and -0.31, for the same two models, respectively, indicating that the model trained on both energies and forces might be better at very large training sets. We note, however, that the current training set is not large enough to confirm this, as this would require a force training set with at least 10 000 to 50 000 training molecules (See figure 3(D), which, in turn, would render this experiment computationally intractable with a kernel matrix of up to around $2\,000\,000 \times 2\,000\,000$ elements.

When the curves are viewed as a function of the total number of training labels (in figure 3(B), the energy-only model requires about 51 times fewer labels to reach the same accuracy as the model trained on both forces and energies. This indicates, that the additional force labels do not aid the model in predicting energies of unseen molecules. This is likely due to the fact that forces can only determine the potential surface up to an integration constant, which can only be learned using energy labels.

Next, in figure 3(C), the MAE predicted force components are plotted for the two models as a function of the number of molecules in the training set. On average, including the forces in the training set reduces the predicted MAE force component by a factor of 3.5. Essentially, while the additional gradient information is not able to improve learning beyond the integration constant, it does help improve the prediction of the relative energy landscape of each potential energy surface, in turn leading to improved gradient predictions, but at the cost of much larger number of training labels. Figure 3(D) displays the same MAE predicted force components as in figure 3(C), but as a function of the total number of training labels. In this case, we observe that the two learning curves are very close-lying, similarly to what was found for the revised MD17 dataset in the previous section. This suggests that for the training sizes investigated herein, one training energy label is worth roughly the same as one training force component label, although the differences in slopes indicate that the model trained on energies and forces might reach superior accuracy at much larger training sizes where the role of the integration constant is diminished.
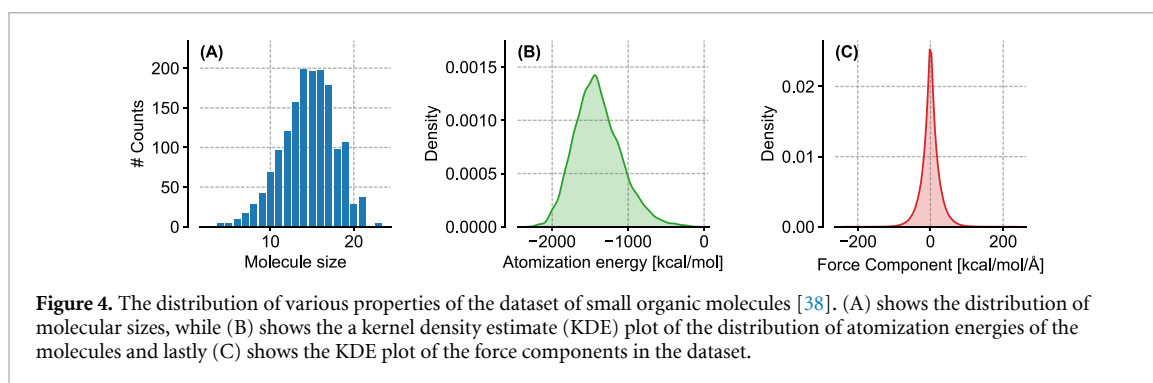
4. Methodology

4.1. Hyperparameter selection and learning curves

All learning curves were generated using nested cross-validation as implemented in `scikit-learn`[46] via the following recipe: First, the datasets were randomized. Secondly, the datasets were divided into 100 folds using random subsampling for Himmielblau's function, while datasets consisting of molecules were randomly divided into 5 folds using the `KFold` class implemented in `scikit-learn`. Next, a grid-search with 4-fold cross-validation within the training set extracted from the fold was used to select the optimal choice of the hyperparameters σ (the kernel width) and λ (regularization strength), in order to avoid overfitting. In order to select hyperparameters that simultaneously work well for both force and energy prediction, the following loss function was used to select these: [15]

$$\mathcal{L} = 0.01 \sum_i \left(U_i - \hat{U}_i \right)^2 + \sum_i \frac{1}{n_i} \|\mathbf{F}_i - \hat{\mathbf{F}}_i\|^2 \tag{19}$$

Here, U_i is the energy of the ith molecule and $\mathbf{F}_i$ and n_i are the force-vector and number of atoms in the same atom, respectively. We acknowledge that the functional form of this loss function is somewhat empirical in nature. However, in contrast with the loss function which is minimized in the regression step, this loss function never minimizes to zero because it is evaluated on out-of-training samples. For this reason it is necessary to balance the contributions of the prediction errors from energies as well as forces, as (i) the energies are far fewer in numbers and (ii) they exist on different scales of units. This particular functional form has previously been shown to exhibit a fair balance between these considerations [15, 22].

In the cases where either force/function derivatives or energies/function values were not included in the training data, the first or the second term was left out, respectively.

Figure 4. The distribution of various properties of the dataset of small organic molecules [38]. (A) shows the distribution of molecular sizes, while (B) shows the a kernel density estimate (KDE) plot of the distribution of atomization energies of the molecules and lastly (C) shows the KDE plot of the force components in the dataset.

For the grid searches for Himmelblau's function, the kernel width was tested in the range $\sigma \in \{0.4, 0.8, 1.6, 3.2, 6.4, 12.8, 25.6\}$ and the regularization strength was tested in the range $\lambda \in \{10^{-13}, 10^{-12}, 10^{-11}, 10^{-10}, 10^{-9}, 10^{-8}\}$. For the molecular datasets (see next section), the grid-searches used values of $\sigma \in \{0.25, 0.5, 1.0, 2.0, 4.0, 8.0, 16.0, 32.0\}$ for the kernel width and $\lambda \in \{10^{-12}, 10^{-11}, 10^{-10}, 10^{-9}, 10^{-8}, 10^{-7}, 10^{-6}\}$ for the regularization strength.

4.2. Software
All machine learning models were implemented in `Python` using the `QML` machine learning package [47]. The packages `Matplotlib` and `Seaborn` were used to plot all figures and `Scikit-Learn` was used to obtain the linear fits to learning curves [46, 48, 49].

4.3. Datasets
This subsection briefly presents the used datasets and their availability.

4.3.1. Revised MD17 dataset
For each of the 10 molecules in the MD17 dataset [20, 21] (aspirin, benzene, ethanol, salicylic acid, malonaldehyde, toluene, naphthalene, uracil, paracetamol and azobenzene), 100,000 structures were randomly selected from the available MD trajectory data. For each of these structures, a single-point force and energy evaluation was carried out at the DFT level. All calculations were performed in ORCA 4.0.1, using the PBE functional and the def2-SVP basis set with the resolution-of-identity (RI) approximation for the Coulomb integrals [50–52]. In order to have a minimal unsystematic error, the keyword `VeryTightSCF` was used to reduce the error from SCF convergence, while the keywords `Grid7` and `NoFinalGrid` were used to utilize the largest standard grid implemented in ORCA. This data has been uploaded to https://dx.doi.org/10.6084/m9.figshare.12672038 and https://dx.doi.org/10.24435/materialscloud:wy-kn along with the indices used for the outer 5-fold cross-validation.

4.3.2. Small organic molecules
This dataset is taken from reference [38] and consists of non-equilibrium conformers of 1,595 small organic molecules with up to 7 atoms of the elements CNO saturated with hydrogen atoms. This data is available for download at https://dx.doi.org/10.6084/m9.figshare.7000280. For each of these structures generated via normal-mode sampling, the atomization energy and corresponding forces were provided at the ωB97xD/6-31G(d) level of theory [53]. Figure 4 shows plots of the distributions of molecular sizes, energies and forces in the dataset.

5. Conclusion and outlook

In this paper we have presented numerical confirmation that the predictive error of machines trained on function derivatives follows a power law, a result that is well-known and to be expected for both neural networks as well as kernel models trained on scalar values. Our numerical results demonstrate this power-law behaviour for two popular and distinct types of molecular datasets, as well as for the Himmelblau's function for which 6 orders of magnitude have been spanned.

For datasets that deal with a single surface, such as Himmelblaus's function and the potential energy surfaces of 10 molecules in the MD17 dataset, we find that including force labels in the set of training labels—in addition to energy labels—leads to an improvement of the predictive error when the same number of training points are used. At the same time, however, we only find negligible differences between models trained on force labels only and models trained on both energy and force labels. Regarding the number of

total training labels (rather than number of molecules in the MD17 dataset), we found that the predictive error of forces and energies was close to identical, regardless of whether the model was trained on energies only, trained on both energies and forces, or trained on forces only.

For the diverse dataset of distinct organic molecules, we find that the prediction error of energies does not improve upon addition of force labels to energy labels in the training set, in contrast to what was observed for Himmelblau's function and the MD17 dataset. Conversely, the predictive error of atomic forces is greatly improved by including forces in the training set. The likely explanation is that geometrical derivative information only helps determine the surface up to an integration constant, corresponding to the atomization energy. This effect is more visible when viewing the learning rate as a function of the number of training labels. In this case, energies are learned with about $20 \times$ less training labels, training on energies only, compared to training on both forces and energies. At the same time, the predictive force error when training on either energies, or forces and energies simultaneously, seems to decay at comparable rates with the number of force labels in these two scenarios. In order to use derivative information to further improve this learning, it would be necessary to use *alchemical* derivatives, i.e. the derivatives of the energy with respect to the atomic charges [40, 54] which form the basis of alchemical perturbation density functional theory [55, 56] These derivatives describe the change in energy as one molecule is alchemically transformed to another molecule and could allow for better interpolation between potential energy surfaces of different molecules in a training set.

We believe that our observations have implications for the generation of new molecular datasets. Here we point out that it is necessary to account for the cost of obtaining the training labels. With this in mind, it is beneficial for use cases of the MD17-type to include forces labels in the training set when the cost of acquisition is less than the cost of 3 N single-point energy calculations on un-correlated samples, where N is the number of atoms in the molecule. This suggests that for DFT-based datasets, it seems very favorable to calculate and report gradients in addition to single-point energies, while for methods with more costly gradients (compared to the energy evaluation) this becomes less favorable.

For energy predictions throughout large datasets used to fit models for general chemistry, i.e. throughout chemical space, such as, for example, the family of ANI datasets [5, 25–27], it seems more valuable to build the most compact model using a compositionally diverse training set with only single-point energies, rather than 'wasting' coefficients by training also on forces for more conformations of the same molecule. If the goal, however, consists of *also* modeling forces, such as is typically the case for relaxing geometries throughout chemical compound space, our results indicate that the addition of forces (if acquisition cost is lower than for energies) in the training set is always beneficial. This conclusion, however, also depends on the requirements for execution speed and the availability of training data: Models trained on force labels can be computationally substantially more expensive to train and execute compared to models on the same number of energy labels, since this often involves the derivative of, for example, a kernel matrix or a neural network. Consequently, in an application scenario where sufficient energy labels are available, it might be best to train on energy labels only, as this enables numerically less complex training models. Considering the prediction times for kernel-based models, it is also much more computationally expensive to evaluate kernel functions placed on derivatives compared to those placed on scalars. If the ultimate goal is to have very fast prediction times for kernel-based models, it seems worthwhile to consider the use of kernel-based force models which do not require the evaluation of second-order kernel derivatives.

All these observations summarize our insights into the design of future data-driven models and their underlying dataset generation. We believe that they are applicable to all branches of machine learning where the goal is to learn multidimensional, differentiable function surfaces.

ORCID iD

Anders S Christensen ⓘ https://orcid.org/0000-0002-7253-6897

References

[1] Bartók A P and Gábor Canyi 2015 Gaussian approximation potentials: A brief tutorial introduction *Int. J. Quantum Chem.* **115** 1051–7

[2] Lorenz S, Gross A and Scheffler M 2004 Representing high-dimensional potential-energy surfaces for reactions at surfaces by neural networks *Chem. Phys. Lett.* **395** 210

[3] Behler J and Parrinello M 2007 Generalized neural-network representation of high-dimensional potential-energy surfaces *Phys. Rev. Lett.* **98** 146401

[4] Behler Jorg 2016 Perspective: machine learning potentials for atomistic simulations *J. Chem. Phys.* **145** 170901

[5] Smith J S, Isayev O and Roitberg A E 2017 ANI-1: An extensible neural network potential with DFT accuracy at force field computational cost *Chem. Sci.* **8** 3192–203

[6] Botu V and Ramprasad R 2015 Learning scheme to predict atomic forces and accelerate materials simulations *Phys. Rev.* B **92** 094306

[7] Botu V and Ramprasad R 2015 Adaptive machine learning framework to accelerate ab initio molecular dynamics *Int. J. Quantum Chem.* **115** 1074–83

[8] Botu V, Batra R, Chapman J and Ramprasad R 2016 Machine learning force fields: Construction, validation and outlook *J. Phys. Chem.* C **121** 511–22

[9] Huan T D, Batra R, Chapman J, Krishnan S, Chen L and Ramprasad R 2017 A universal strategy for the creation of machine learning-based atomistic force fields *Npj Comput. Mater.* **3** 37

[10] Zhenwei Li, Kermode J R and Alessandro D V 2015 Molecular dynamics with on-the-fly machine learning of quantum-mechanical forces *Phys. Rev. Lett.* **114** 096405

[11] Gubaev K, Podryabinkin E V and Shapeev A V 2018 Machine learning of molecular properties: Locality and active learning *J. Chem. Phys.* **148** 241727

[12] Thompson A P, Swiler L P, Trott C R, Foiles S M and Tucker G J 2015 Spectral neighbor analysis method for automated generation of quantum-accurate interatomic potentials *J. Comput. Phys.* **285** 316–30

[13] Glielmo A, Sollich P and Alessandro D V 2017 Accurate interatomic force fields via machine learning with covariant kernels *Phys. Rev.* B **95** 214302

[14] Glielmo A, Zeni C and Alessandro D V 2018 Efficient nonparametric *n*-body force fields from machine learning *Phys. Rev.* B **97** 184307

[15] Schütt K T, Sauceda H E, Kindermans P-J, Tkatchenko A and Müller K-R 2018 SchNet–A deep learning architecture for molecules and materials *J. Chem. Phys.* **148** 241722

[16] Schütt K T, Kessel P, Gastegger M, Nicoli K A, Tkatchenko A and Müller K-R 2019 SchNetPack: a deep learning toolbox for atomistic systems *J. Chem. Theory Comput.* **15** 448–55

[17] Grisafi A, Wilkins D M, Csányi Gabor and Ceriotti M 2018 Symmetry-adapted machine learning for tensorial properties of atomistic systems *Phys. Rev. Lett.* **120** 036002

[18] Zhang L, Han J, Wang H, Car R and Weinan E 2018 Deep potential molecular dynamics: A scalable model with the accuracy of quantum mechanics *Phys. Rev. Lett.* **120** 143001

[19] Unke O T and Meuwly M 2019 PhysNet: a neural network for predicting energies, forces, dipole moments and partial charges *J. Chem. Theory Comput.* **15** 3678–93

[20] Chmiela S, Tkatchenko A, Sauceda H E, Poltavsky I, Schütt K T and Müller K-R 2017 Machine learning of accurate energy-conserving molecular force fields *Sci. Adv.* **3** e1603015

[21] Chmiela S, Sauceda H E, Müller K-R and Tkatchenko A 2018 Towards exact molecular dynamics simulations with machine-learned force fields *Nat. Commun.* **9** 3887

[22] Christensen A S, Bratholm L A, Faber F A and von Lilienfeld O A 2020 Fchl revisited: Faster and more accurate quantum machine learning *J. Chem. Phys.* **152** 044107

[23] Lubbers N, Smith J S and Barros K 2018 Hierarchical modeling of molecular energies using a deep neural network *J. Chem. Phys.* **148** 241715

[24] Schütt K T, Arbabzadah F, Chmiela S, Müller K R and Tkatchenko A 2017 Quantum-chemical insights from deep tensor neural networks *Nat. Comm.* **8** 13890

[25] Smith J S, Isayev O and Roitberg A E 2017 ANI-1, A data set of 20 million calculated off-equilibrium conformations for organic molecules *Sci. Data* **4** 170193

[26] Smith J S, Nebgen B, Lubbers N, Isayev O and Roitberg A E 2018 Less is more: Sampling chemical space with active learning *J. Chem. Phys.* **148** 241733

[27] Smith J S 2020 The ani-1ccx and ani-1x data sets, coupled-cluster and density functional theory properties for molecules *Sci. Data* **7** 134

[28] Vapnik V 2013 *The Nature of Statistical Learning Theory* (Berlin: Springer Science & Business Media)

[29] von Lilienfeld O A 2018 Quantum machine learning in chemical compound space *Angew. Chem. Int. Ed.* **57** 4164

[30] Cortes C, Jackel L D, Solla S A, Vapnik V and Denker J S 1994 Learning curves: Asymptotic values and rate of convergence *Advances in Neural Information Processing Systems* (San Mateo, CA: Morgan Kaufmann Publishers) pp 327–334

[31] Müller K R, Finke M, Murata N, Schulten K and Amari S 1996 A numerical study on learning curves in stochastic multilayer feedforward networks *Neural Comp.* **8** 1085

[32] Himmelblau D M 1972 *Applied Nonlinear Programming* (New York: McGraw-Hill)

[33] Hansen K, Montavon G, Biegler F, Fazli S, Rupp M, Scheffler M, von Lilienfeld O A, Tkatchenko A and Müller K-R 2013 Assessment and validation of machine learning methods for predicting molecular atomization energies *J. Chem. Theory Comput.* **9** 3404–19

[34] Solak E, Murray-Smith R, Leithead W E, Leith D J and Rasmussen C E 2003 *Advances in Neural Information Processing Systems 15* ed Becker S, Thrun S and Obermayer K (Cambridge, MA: MIT Press) pp 1057–1064 *Advances in Neural Information Processing Systems 15*

[35] Rasmussen C E and Williams C K I 2006 *Gaussian Processes for Machine Learning* Dietterich T (Cambridge: MIT Press) Www.gaussianprocess.org

[36] Mones L, Bernstein N and Csányi Gabor 2016 Exploration, sampling and reconstruction of free energy surfaces with Gaussian process regression *J. Chem. Theory Comput.* **12** 5100–10 PMID: 27598684

[37] Bartók A P, Payne M C, Kondor R and Csányi Gabor 2010 Gaussian approximation potentials: The accuracy of quantum mechanics, without the electrons *Phys. Rev. Lett.* **104** 136403

[38] Christensen A S, Faber F A and von Lilienfeld O A 2019 Operators in quantum machine learning: Response properties in chemical space *J. Chem. Phys.* **150** 064105

[39] Mathias S 2015 2015 A kernel-based learning method for an efficient approximation of the high-dimensional Born-Oppenheimer potential energy surface *Master's thesis* Mathematisch-Naturwissenschaftliche Fakultät der Rheinischen Friedrich-Wilhelms-Universität Bonn Germany http://wissrech.ins.uni-bonn.de/teaching/master/masterthesis_mathias_revised.pdf

[40] von Lilienfeld O A 2013 First principles view on chemical compound space: gaining rigorous atomistic control of molecular properties *Int. J. Quantum Chem.* **113** 1676–89

[41] Hansen K, Biegler F, Ramakrishnan R, Wiktor Pronobis O von Lilienfeld A, Müller K-R and Tkatchenko A 2015 *J. Phys. Chem. Lett.* **6** 2326–31 Machine learning predictions of molecular properties: accurate many-body potentials and nonlocality in chemical space

[42] Faber F A, Christensen A S, Huang B and von Lilienfeld O A 2018 Alchemical and structural distribution based representation for universal quantum machine learning *J. Chem. Phys.* **148** 241717

[43] Faber F A, Christensen A S, Huang B and von Lilienfeld O A 2018 Alchemical and structural distribution based representation for universal quantum machine learning *J. Chem. Phys.* **148** 241717

[44] Barker J, Bulin J, Hamaekers J and Mathias S 2016 Localized Coulomb Descriptors for the Gaussian Approximation Potential Springer Berlin 25-42

[45] Pozdnyakov S N, Willatt M J, Bartók A P, Ortner C, Csányi Gabor and Ceriotti M 2020 On the Completeness of Atomic Structure Representations **125** 166001

[46] Pedregosa F *hboxet al* 2011 Scikit-learn: Machine learning in Python *J. Mach. Learn. Res.* **12** 2825–30

[47] Christensen A S, Faber F A, Huang B, Bratholm L A, Tkatchenko A, Müller K-R and von Lilienfeld O A 2017 *Qml: A Python Toolkit for Quantum Machine Learning* (https://github.com/qmlcode/qml)

[48] Hunter J D 2007 Matplotlib: A 2d graphics environment *Computing Sci. Eng.* **9** 90–5

[49] Waskom M *hboxet al* 2017 mwaskom/seaborn: v0.8.1 (september 2017) September

[50] Frank N 2017 Software update: the orca program system, version 4.0 *Wiley. Interdiscip. Rev. Comput. Mol. Sci.* **8** e1327

[51] Perdew J P, Burke K and Ernzerhof M 1996 Generalized gradient approximation made simple *Phys. Rev. Lett.* **77** 3865

[52] Weigend F and Ahlrichs R 2005 Balanced basis sets of split valence, triple zeta valence and quadruple zeta valence quality for H to Rn: design and assessment of accuracy *Phys. Chem. Chem. Phys.* **7** 3297

[53] Chai J-D and Head-Gordon M 2008 Long-range corrected hybrid density functionals with damped atom-atom dispersion corrections *Phys. Chem. Chem. Phys.* **10** 6615

[54] von Lilienfeld O A, Lins R and Rothlisberger U 2005 Variational particle number approach for rational compound design *Phys. Rev. Lett.* **95** 153002

[55] von Rudorff G F and von Lilienfeld O A 2020 Alchemical perturbation density functional theory *Phys. Rev. Res.* **2** 023220

[56] von Rudorff G F and von Lilienfeld O A 2020 Rapid and accurate molecular deprotonation energies from quantum alchemy *Phys. Chem. Chem. Phys.* **22** 10519–25

[57] Oliphant T 2019 NumPy: A guide to NumPy USA: Trelgol Publishing 2006– [Online; accessed July]

[58] OpenMP Architecture Review Board 2008 OpenMP application program interface version 3.0