SchNetPack: A Deep Learning Toolbox For Atomistic Systems

K. T. Schütt,*,† P. Kessel,† M. Gastegger,† K. A. Nicoli,† A. Tkatchenko,*,‡ and K.-R. Müller*,†,§,¶

†Machine Learning Group, Technische Universität Berlin, 10587 Berlin, Germany

‡Physics and Materials Science Research Unit, University of Luxembourg, L-1511 Luxembourg, Luxembourg

§Department of Brain and Cognitive Engineering, Korea University, Anam-dong, Seongbuk-gu, Seoul 02841, South Korea

¶Max-Planck-Institut für Informatik, Saarbrücken, Germany

S *Supporting Information*

ABSTRACT: SchNetPack is a toolbox for the development and application of deep neural networks that predict potential energy surfaces and other quantum-chemical properties of molecules and materials. It contains basic building blocks of atomistic neural networks, manages their training, and provides simple access to common benchmark datasets. This allows for an easy implementation and evaluation of new models. For now, SchNetPack includes implementations of (weighted) atom-centered symmetry functions and the deep tensor neural network SchNet, as well as ready-to-use scripts that allow one to train these models on molecule and material datasets. Based on the PyTorch deep learning framework, SchNetPack allows one to efficiently apply the neural networks to large datasets with millions of reference calculations, as well as parallelize the model across multiple GPUs. Finally, SchNetPack provides an interface to the Atomic Simulation Environment in order to make trained models easily accessible to researchers that are not yet familiar with neural networks.
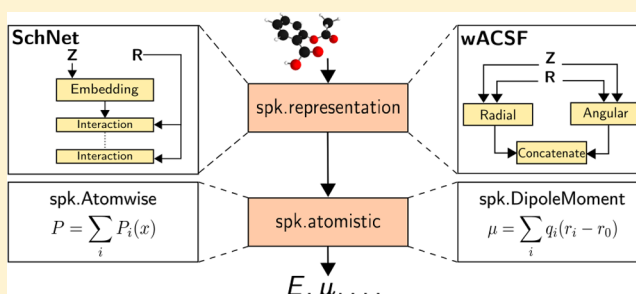
1. INTRODUCTION

One of the fundamental aims of modern quantum chemistry, condensed matter physics, and materials science is to numerically determine the properties of molecules and materials. Unfortunately, the computational cost of accurate calculations prove prohibitive when it comes to large-scale molecular dynamics simulations or the exhaustive exploration of the vast chemical space. Over the last years, however, it has become clear that machine learning is able to provide accurate predictions of chemical properties at significantly reduced computational costs. Conceptually, this is achieved by training a machine learning model to reproduce the results of reference calculations given the configuration of an atomistic system. Once trained, predicting properties of other atomistic systems is generically inexpensive and has been shown to be sufficiently accurate for a range of applications.[1−18]

A common subclass of machine learning models for quantum chemistry are *atomistic neural networks*. There exist various architectures of these models, which can be broadly split into two categories: descriptor-based models, which take a predefined representation of the atomistic system as input,[19−24] and end-to-end architectures, that learn a representation directly from atom types and positions.[25−28]

SchNetPack provides a unified framework for both categories of neural networks. While we plan to support more architectures in the future, SchNetPack currently includes implementations for SchNet,[15,27] an end-to-end continuous convolution architecture, as well as Behler−

Parrinello networks, which are based on atom-centered symmetry functions (ACSF),[19,29] and an extension thereof, which uses weighted atom-centered symmetry functions (wACSF).[24]

SchNetPack furthermore contains functionality for accessing popular benchmark data sets, training neural networks on (multiple) GPUs to predict a variety of chemical properties. It is built in an extensible manner and is implemented using the PyTorch deep learning framework.

The remainder of the paper is structured as follows. In Section 2, we present how models in SchNetPack are structured and briefly review (w)ACSF and SchNet representations. Section 3 outlines how SchNetPack manages the training process for atomistic neural networks and gives an overview of the integrated datasets. Section 4 summarizes details about the implementation, while Sections 5 and 6 provide code examples for training an atomistic neural network and calculating a power spectrum using the interface to the Atomic Simulation Environment (ASE).[32] Section 7 presents results of SchNetPack on standard benchmarks, before we conclude and give an outlook on future extensions.

2. MODELS

Models in SchNetPack have two principle components: *representation* and *prediction* blocks (see Figure 1). The former
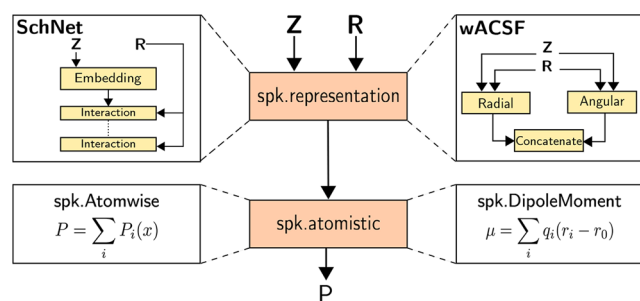
Figure 1. Basic building blocks of a model predicting the property P from the positions R and atomic numbers $\mathbf{Z}$ of the atomistic system. We use the abbreviation `spk` for the `schnetpack` package. All *representation* and *prediction* blocks are collected in the `spk.representation` and `spk.atomistic` package, respectively. The right and left panels illustrate various choices for these building blocks.

takes the configuration of the atomistic system as an input and generates feature vectors describing each atom in its chemical environment. The latter uses these atom-wise representations to predict the desired properties of the atomistic system. The only difference between descriptor-based and end-to-end architectures is whether the representation block is fixed or learned from data. In the following two sections, we will explain the possible choices for these components in detail.

2.1. Representations. An atomistic system containing n atoms can be described by its atomic numbers $\mathbf{Z} = (Z_1, ..., Z_n)$ and positions $R = (\mathbf{r}_1, ..., \mathbf{r}_n)$. The interatomic distances are given as $r_{ij} = \|\mathbf{r}_i - \mathbf{r}_j\|$. In the following, we will briefly describe the currently implemented representations, i.e., (w)ACSF[24] and SchNet.[27] For further details, refer to the original publications.

2.1.1. (w)ACSF. Behler−Parrinello network potentials[19] have proven very useful for systems as diverse as small molecules, metal and molecular clusters, bulk materials, surfaces, water and solid−liquid interfaces (for a recent review, see ref 30). Because of this impressive number of applications, Behler−Parrinello networks are now firmly established as a highly successful neural network architecture for atomistic systems.

For these networks, so-called atom-centered symmetry functions (ACSFs) form the representation of the atomistic system. Contrary to the approach taken by SchNet, where features are learned from the data, ACSFs must be determined before training. Hence, using symmetry functions can be advantageous in situations where the available training data is insufficient to learn suitable representations in an end-to-end fashion. On the other hand, introducing rigid hand-crafted features might reduce the generality of the model. In the following, we will briefly review ACSFs and a variant called weighted ACSFs, or wACSFs for short. We refer to refs 29 and 24 for a more-detailed discussion.

ACSFs describe the local chemical environment around a central atom via a combination of radial and angular distribution functions.

2.1.1.1. Radial Symmetry Functions. Radial ACSF descriptors take the following form:

$$G_{i,\alpha}^{\text{rad}} = \sum_{j \neq i}^{N} g(Z_j) e^{-\gamma_\alpha (r_{ij} - \mu_\alpha)^2} f(r_{ij})$$

(1)

where i is the central atom and the sum runs over all neighboring atoms j. γ_α and μ_α are parameters that modulate the widths and centers of the Gaussians. Typically, a set of n_{rad} radial symmetry functions with different parameter combinations $\alpha \in \{1, ..., n_{\text{rad}}\}$ are used. In SchNetPack, suitable γ_α and μ_α are determined automatically via an equidistant grid between zero and a spacial cutoff r_c, adopting the empirical parametrization strategy detailed in ref 24.

A cutoff function f ensures that only atoms close to the central atom i enter the sum, and this function is given by the following expression:

$$f(r_{ij}) = \begin{cases} \dfrac{1}{2}\left(\cos\left(\dfrac{\pi r_{ij}}{r_c}\right) + 1\right) & \text{if } r_{ij} \leq r_c \\ 0 & \text{else} \end{cases}$$

(2)

For convenience, we will use the notation $f_{ij} = f(r_{ij})$ in the following. Finally, $g(Z_j)$ is an element-dependent weighting function. In ACSFs, $g(Z_j)$ takes the form

$$g(Z_j) = \delta_{Z_j, Z_a} = \begin{cases} 1 & \text{if } Z_j = Z_a \\ 0 & \text{else} \end{cases}$$

(3)

Hence, radial ACSFs are always defined between the central atom and a neighbor belonging to a specific chemical element.

2.1.1.2. Angular Symmetry Functions. Information about the angles between atoms are encoded by the n_a angular symmetry functions

$$G_{i,\alpha}^{\text{ang}} = 2^{1-\zeta_\alpha} \sum_{j \neq i, k > j}^{N} g(Z_j, Z_k)(1 + \lambda \theta_{ijk})^{\zeta_\alpha}$$
$$\times \exp\left[-\gamma_\alpha (r_{ij}^2 + r_{ik}^2 + r_{jk}^2)\right] f_{ij} f_{ik} f_{jk}$$

(4)

where θ_{ijk} is the angle spanned between atoms i, j, and k. The parameter λ takes the values $\lambda = \pm 1$, which shifts the maximum of the angular terms between 0 and π. The variable ζ_α is a hyperparameter controlling the width around this maximum. γ_α once again controls the width of the Gaussian functions. As with radial ACSFs, a set of n_{ang} angular functions differing in their parametrization patterns $\alpha \in \{1, ..., n_{\text{ang}}\}$ is chosen to describe the local environment. For angular ACSFs, the weighting function $g(Z_k, Z_j)$ can be expressed as

$$g(Z_k, Z_j) = \frac{1}{2}(\delta_{Z_j, Z_a} \delta_{Z_k, Z_b} + \delta_{Z_j, Z_b} \delta_{Z_k, Z_a})$$

(5)

which counts the contributions of neighboring atoms j and k belonging to a specific pair of elements (e.g., O−H or O−O).

Because of the choice of g, ACSFs always are defined for pairs (radial) or triples (angular) of elements and at least one parametrized function $G_{i,\alpha}$ must be provided for each of these combinations. As a consequence, the number of ACSFs grows quadratically with the number of different chemical species. This can lead to an impractical number of ACSFs for systems containing more than four elements (e.g., QM9).

Recently, alternative weighting functions have been proposed that circumvent the above issue. In these so-called weighted ACSFs (wACSFs), the radial weighting function is chosen as $g(Z_j) = Z_j$ while the angular function is set to $g(Z_k, Z_j) = Z_k Z_j$. Through this simple reparametrization, the number of required symmetry functions becomes independent of the actual number of elements present in the system, leading

to more compact descriptors. SchNetPack uses wACSFs as the standard descriptor for Behler−Parrinello potentials.

Irrespective of the choice for the weighing g, both radial and angular symmetry functions are concatenated as a final step to form the representation for the atomistic system, i.e.,

$$X_i = (G_{i,1}^{\text{rad}}, \, ... \, G_{i,n_{\text{rad}}}^{\text{rad}}, \, G_{i,1}^{\text{ang}}|_{\lambda=\pm 1}, \, ..., \, , \, G_{i,n_{\text{ang}}}^{\text{ang}}|_{\lambda=\pm 1}) \tag{6}$$

This representation of X_i can then serve as input for prediction block of the atomistic network.

2.1.2. SchNet. SchNet is an end-to-end deep neural network architecture based on continuous-filter convolutions.[15,27] It follows the deep tensor neural network framework,[25] i.e., atom-wise representations are constructed by starting from embedding vectors that characterize the atom type before introducing the configuration of the system by a series of interaction blocks.

Convolutional layers in deep learning usually act on discretized signals such as images. Continuous-filter convolutions are a generalization thereof for input signals that are not aligned on a grid, such as atoms at arbitrary positions. Contrary to (w)ACSF networks which are based on rigid hand-crafted features, SchNet adapts the representation of the atomistic system to the training data. More precisely, SchNet is a multilayer neural network that consists of an embedding layer and several interaction blocks, as shown in the top left panel of Figure 1. We describe its components in greater detail in the following subsections.

2.1.2.1. Atom Embeddings. Using an embedding layer, each atom type Z_i is represented by feature vectors $\mathbf{x}_i^0 \in \mathbb{R}^F$, which we collect in a matrix $X^0 = (\mathbf{x}_1^0, ..., \mathbf{x}_n^0)$. The feature dimension is denoted by F. The embedding layer is initialized randomly and adapted during training. In all other layers of SchNet, atoms are described analogously and we denote the features of layer l by $X^l = (\mathbf{x}_1^l, ..., \mathbf{x}_n^l)$ with $\mathbf{x}_i^l \in \mathbb{R}^F$.

2.1.2.2. Interaction Blocks. Using the features X^l and positions R, this building block computes interactions which additively refine the previous representation analogue to ResNet blocks.[31] To incorporate the influence of neighboring atoms, continuous-filter convolutions are applied, which are defined as follows:

$$\mathbf{x}_i^{l+1} = (X^l * W^l) \equiv \sum_{j \in \text{nbh}(i)} \mathbf{x}_j^l \odot W^l(r_{ij}) \tag{7}$$

We use the symbol "$\odot$" to denote element-wise multiplication, and nbh(i) are the neighbors of atom i. In particular, for larger systems, it is recommended to introduce a radial cutoff. For our experiments, we use a distance cutoff of 5 Å.

Here, the filter is not a parameter tensor as in standard convolutional layers, but a filter-generating neural network $W^l: \mathbb{R} \to \mathbb{R}^F$, which maps atomic distances to filter values. The filter generator takes atom positions expanded on a grid of radial basis functions that are closely related to the radial symmetry functions (eq 1) of (w)ACSF. For its precise architecture, we refer to the original publications.[15,27]

Several atom-wise layers, i.e., fully connected layers, which are defined as

$$\mathbf{x}_i^{l+1} = W^l \mathbf{x}_i^l + \mathbf{b}^l \tag{8}$$

are applied to each atom i separately and recombine the features within each atom representation. Note that the weights W^l and biases $\mathbf{b}^l$ are independent of i and, therefore,

are the same for all atom features $\mathbf{x}_i^l$. Thus, the number of parameters of atom-wise layers is independent of the number of atoms n.

In summary, SchNet obtains a latent representation of the atomistic system by first using an embedding layer to obtain features X^0. These features are then processed by L interaction blocks, which results in the latent representation X^L, which can be passed to the prediction block. We will sketch the possibilities for the architectures of these prediction blocks in the following section.

2.2. Prediction Blocks. As discussed in the last sections, both SchNet and (w)ACSF provide representations X_i with $i \in \{1, ..., n\}$ for an atomistic system with n atoms. These representations are then processed by a prediction block to obtain the desired properties of the atomistic system. There are various choices for prediction blocks, depending on the property of interest. Usually, prediction blocks consist of several atom-wise layers (eq 8) with nonlinearities, which reduce the feature dimension, followed by a property-dependent aggregation across atoms.

The most common choice are Atomwise prediction blocks, which express a desired molecular property P as a sum of atom-wise contributions:

$$P = \sum_{i=1}^n p(\mathbf{x}_i) \tag{9}$$

While this is a suitable model for extensive properties such as the energy, intensive properties, which do not grow with the number of atoms n of the atomistic system, are instead expressed as the average over contributions.

Atomwise prediction blocks are suitable for many properties; however, property-specific prediction blocks may be used to incorporate prior knowledge into the model. The DipoleMoment prediction block expresses the dipole moment (μ) as

$$\mu = \sum_{i=1}^n q(\mathbf{x}_i)(\mathbf{r}_i - \mathbf{r}_0) \tag{10}$$

where $q: \mathbb{R}^F \to \mathbb{R}$ can be interpreted as latent atomic charges and $\mathbf{r}_0$ denotes the center of mass of the system.

The ElementalAtomwise prediction block is different from the Atomwise prediction block in that, instead of applying the same network to all the atom features X_i, it uses separate networks for different chemical elements. This is particularly useful for (w)ACSF representations. Analogously, the ElementalDipoleMoment is defined for the dipole moment.

3. DATA PIPELINE AND TRAINING

One of the main aims of SchNetPack is to accelerate the development and application of atomistic neural networks. To this end, SchNetPack contains several classes that provide access to standard benchmark datasets and manage the training process. Figure 2 summarizes this.

The dataset classes automatically download the relevant data, if not already present on disk, and use the standard ASE package[32] to store them in an SQLite database. In particular, this means that we use the conventions and units of the ASE package in SchNetPack, e.g., energies and lengths are in units of eV and Å, respectively. Currently, SchNetPack includes the following dataset classes:
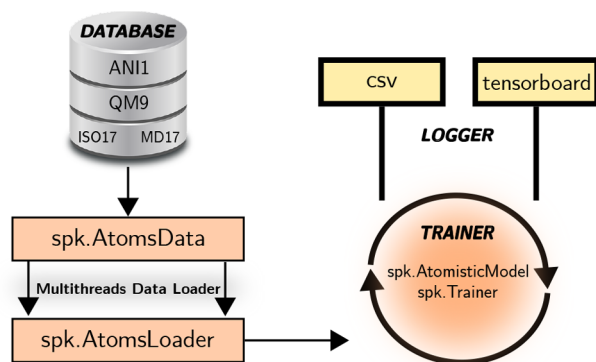
Figure 2. Setup for training models in SchNetPack. Note that we denote the various choices for dataset classes mentioned in Section 3 by their common `spk.AtomsData` base class.

- `schnetpack.datasets.QM9`: class for the QM9 dataset[33,34] for 133 885 organic molecules, with up to 9 heavy atoms from C, O, N, and F.
- `schnetpack.datasets.ANI1`: functionality to access the ANI-1 dataset,[35] which consists of more than 20 million conformations for 57 454 small organic molecules from C, O, and N.
- `schnetpack.datasets.ISO17`: class for ISO17 data set[25,27,34] for molecular dynamics of $C_7O_2H_{10}$ isomers. It contains 129 isomers with 5000 conformational geometries and their corresponding energies and forces.
- `schnetpack.datasets.MD17`: class for MD17 dataset[10,25] for molecular dynamics of small molecules containing molecular forces.
- `schnetpack.datasets.MaterialsProject`: provides access to the Materials Project[36] repository of bulk crystal containing atom types ranging across the entire periodic table up to $Z = 94$.

We also provide a `AtomsLoader` class for feeding a model with (a subset of) a dataset during training using multiple threads. This class also calculates relevant statistics, such as mean and standard deviation.

For convenience, a `Trainer` class is included in SchNetPack, which manages the training process of the model. This class evaluates the model's performance on a validation set, provides functionality for early stopping and various learning rate schedules as well as checkpointing and logging. For the latter, one can choose between csv files and Tensorboard,[37] which is a powerful web-based visualization interface. SchNetPack supports training on multiple GPUs for which we use the standard PyTorch implementation.

As we will show in the example discussed in Section 5, the classes presented in this section allow us to efficiently train atomistic neural networks and evaluate their performance using a very compact amount of code.

4. IMPLEMENTATION DETAILS

SchNetPack is implemented in Python using the PyTorch (≥ 0.4) deep learning library.[38] Calculations that do not require automatic differentiation are performed using Numpy.[39] SchNetPack is tightly integrated with the Atomic Simulation Environment (ASE),[32] which is used to persist configurations of atomistic systems. We also provide an interface to the ASE calculator class, which allows one to easily incorporate

SchNetPack models into ASE workflows, such as performing molecular dynamics. Logging the training progress to Tensorboard[37] is facilitated by tensorboardX.[40] Some of the datasets come in the HDF5 binary file format, which we parse with the h5py package.[41] SchNetPack can be easily installed using pip. [To install run the following command: pip install schnetpack.] The code for SchNetPack can be found on GitHub. [Code can be found here: https://github.com/atomistic-machine-learning/schnetpack.]

5. EXAMPLE: TRAINING IN SCHNETPACK

Chart 1 is a minimal example of how to train a model with SchNet representation to predict the total energy U_0 on QM9.

Chart 1. Minimal Code Example for Training a SchNet Model on the QM9 Dataset with SchNetPack

```python
1  import schnetpack as spk
2  import schnetpack.atomistic as atm
3  import schnetpack.representation as rep
4  import torch
5  from torch.optim import Adam
6  import torch.nn.functional as F
7  from schnetpack.datasets import *
8
9  # load qm9 dataset and download if necessary
10 data = QM9("qm9/", properties=[QM9.U0])
11
12 # split in train and val
13 train, val, test = data.create_splits(10000,
                                         1000)
15 loader = spk.AtomsLoader(train,
16                          batch_size=100,
17                          num_workers=4)
18 val_loader = spk.AtomsLoader(val)
19
20 # create model
21 reps = rep.SchNet()
22 output = atm.Atomwise()
23 model = atm.AtomisticModel(reps, output)
24
25 # create trainer
26 opt = Adam(model.parameters(), lr=1e-4)
27 loss = lambda b,p: F.mse_loss(p["y"],b[QM9.U0])
28 trainer = spk.Trainer("output/", model, loss,
29                       opt, loader, val_loader)
30
31 # start training
32 trainer.train(torch.device("cpu"))
```

Training and validation sets with 10k and 1k data points are used, and the data are loaded asynchronously, using four worker threads.

In order to train on a different dataset, one must only change line 10 in Chart 1. In the example of ANI-1, it will read

```python
data = ANI1("ani1/", properties=[ANI1.energy])
```

Similarly, one can straightforwardly change the representation to wACSF by replacing line 21 by

```python
reps = rep.BehlerSFBlock()
```

In this case, however, it is advantageous to use a `ElementalAtomwise` output network by changing line 22 to

```
output = atm.ElementalAtomwise(reps.n_symfuncs)
```

These examples can also be found in the SchNetPack source directory in the examples subdirectory.

6. EXAMPLE: SCHNETPACK FOR CHEMISTS

In addition to the above features, SchNetPack provides an interface to the ASE `Calculator` class. This makes it possible to use SchNetPack models with the calculation tools available in ASE, such as geometry optimization, normal-mode analysis, and molecular dynamics simulations.

The ASE interface is provided via the `AseDriver` class in the `molecular_dynamics` module.

Chart 2 shows an example on how trained models are loaded into the calculator and used for computation. For convenience,

Chart 2. Minimal Code Example for Performing ASE Calculations with a Trained SchNetPack Model Stored in `model_directory`

```
1  import schnetpack.molecular_dynamics as md
2
3  # Load trained model
4  model = md.load_model(model_directory)
5
6  ml_calculator = md.AseInterface(
7      path_to_molecule, model,
8      simulation_directory
9      )
10
11 # Optimize structure
12 ml_calculator.optimize()
13
14 # Compute numerical normal models
15 ml_calculator.compute_normal_modes()
16
17 # Setup and run molecular dynamics
18 ml_calculator.init_md()
19 ml_calculator.run_md()
```

SchNetPack provides the script `schnetpack_molecular_dynamics.py`, which can be used to perform various simulations out of the box. To demonstrate the above features, SchNetPack was used to predict the power spectrum of the keto form of malondialdehyde via molecular dynamics simulations (shown in Figure 3). The machine learning models are able to reproduce the peak positions accurately, even when trained on the smaller dataset, demonstrating the efficacy of the force training procedure. Particularly impressive are the fine details observed in the spectra. For example, the two models are able to resolve the structure of the peak at 1700 cm^{-1}, which is due to the symmetric and asymmetric stretching vibrations of the two carbonyl groups.

SchNet simulations of malondialdehyde require ~ 11 ms per time step on a Tesla P100 GPU. This corresponds to an acceleration of ~ 3 orders of magnitude, compared to the original electronic structure reference computations. In the present setup, Behler−Parrinello networks show a comparable performance to SchNet, which indicates that both models do not yet exhaust the full capacity of the GPU for molecules of this size. It can be expected that ACSF-based models are more efficient when simulating larger systems and also when using CPUs instead of GPUs.

7. RESULTS

In this section, we present results on QM9, ANI-1, MD17, and Material Project datasets obtained with SchNetPack. A summary of the test set performance of both Behler−Parrinello (ACSF and wACSF) and SchNet models can be found in Table 1. The reported results are the average of three models trained on different splits of the same size. The Python scripts with which we obtained these results, using a Tesla-P100 GPU, can be found in the scripts subdirectory of SchNetPack.

Although Behler−Parrinello networks produce reliable results for a wide range of experiments, they are consistently outperformed by the SchNet architecture. Because of its end-to-end nature, SchNet is able to infer efficient molecular representations in a data-driven fashion, which leads to an improved flexibility, compared to the rigid handcrafted features used in Behler−Parrinello potentials (ACSF and wACSFs). The expressive power of SchNet models is enhanced further by their deep architecture, compared to the shallow atomistic
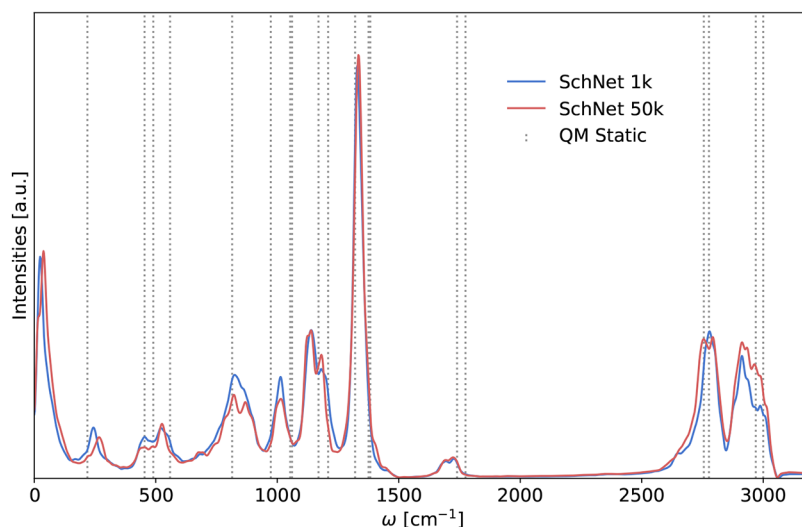

Figure 3. Power spectra of malondialdehyde at 300 K, using SchNets trained on 1000 and 50 000 data points taken from the MD17 database. The harmonic normal mode vibrations obtained with the electronic structure reference are shown in gray.

Table 1. Summary of Performance on Test Set[a]

property	unit	model	MAE	RMSE	time
		Dataset: Malondialdehyde (N = 1k)			
energy	kcal mol^{-1}	SchNet	0.08	0.11	2.5 h
energy	kcal mol^{-1}	ACSF	0.30	0.40	0.6 h
energy	kcal mol^{-1}	wACSF	1.16	1.52	0.6 h
atomic forces	kcal mol^{-1} Å^{-1}	SchNet	0.13	0.16	2.5 h
atomic forces	kcal mol^{-1} Å^{-1}	ACSF	1.08	1.59	0.6 h
atomic forces	kcal mol^{-1} Å^{-1}	wACSF	3.27	4.53	0.6 h
		Dataset: Malondialdehyde (N = 50k)			
energy	kcal mol^{-1}	SchNet	0.07	0.09	13.5 h
energy	kcal mol^{-1}	ACSF	0.09	0.11	6 h
energy	kcal mol^{-1}	wACSF	0.69	0.88	6 h
atomic forces	kcal mol^{-1} Å^{-1}	SchNet	0.05	0.09	13.5 h
atomic forces	kcal mol^{-1} Å^{-1}	ACSF	0.26	0.42	6 h
atomic forces	kcal mol^{-1} Å^{-1}	wACSF	1.84	2.51	6 h
		Dataset: Acetylsalicylic Acid (N = 1k)			
energy	kcal mol^{-1}	SchNet	0.38	0.52	2.5 h
energy	kcal mol^{-1}	ACSF	0.79	1.03	0.7 h
energy	kcal mol^{-1}	wACSF	2.11	2.69	0.7 h
atomic forces	kcal mol^{-1} Å^{-1}	SchNet	1.17	1.68	2.5 h
atomic forces	kcal mol^{-1} Å^{-1}	ACSF	1.92	2.75	0.7 h
atomic forces	kcal mol^{-1} Å^{-1}	wACSF	4.80	6.81	0.7 h
		Dataset: Acetylsalicylic acid (N = 50k)			
energy	kcal mol^{-1}	SchNet	0.11	0.14	2 d, 11.5 h
energy	kcal mol^{-1}	ACSF	0.40	0.53	1 d, 6 h
energy	kcal mol^{-1}	wACSF	1.20	2.69	1 d, 6 h
atomic forces	kcal mol^{-1} Å^{-1}	SchNet	0.14	0.19	2 d, 11.5 h
atomic forces	kcal mol^{-1} Å^{-1}	ACSF	0.88	1.26	1 d, 6 h
atomic forces	kcal mol^{-1} Å^{-1}	wACSF	2.31	3.14	1 d, 6 h
		Dataset: QM9 (N = 110k)			
U_0	kcal mol^{-1}	SchNet	0.26	0.54	12 h
U_0	kcal mol^{-1}	ACSF	0.49	0.92	8 h
U_0	kcal mol^{-1}	wACSF	0.43	0.81	6 h
dipole moment	Debye	SchNet	0.020	0.038	13 h
dipole moment	Debye	ACSF	0.064	0.100	8 h
dipole moment	Debye	wACSF	0.064	0.095	8 h
		Dataset: ANI-1 (N = 10.1M)			
energy	kcal mol^{-1}	SchNet	0.55	0.89	9 d, 7 h[b]
		Dataset: ANI-1 (N = 19.8M)			
energy	kcal mol^{-1}	SchNet	0.47	0.77	12 d, 15 h[c]
		Dataset: Materials Project (N = 62k)			
formation energy	eV/atom	SchNet	0.041	0.088	1 d, 14 h

[a]N is used to denote the size of the combined training set and validation set. [b]Four Tesla P100 GPUs were used for data-parallel training. [c]Two Tesla P100 GPUs were used for data-parallel training.

networks used in Behler−Parrinello models. These features are also advantageous for learning molecular forces for which derivatives of the energy prediction are required for training. A good example are the results obtained for the molecules malonaldehyde and acetylsalicylic acid taken from the MD17 dataset. Here, SchNet outperforms the other models, even on small training sets. SchNet achieves chemically accurate performance for datasets containing a wealth of different molecular configurations (ANI-1), as well as for compounds incorporating a wide range of chemical elements, demonstrating its high utility.

The prime advantage of Behler−Parrinello models is their reduced computational cost, compared to SchNet, which is expected to be beneficial, e.g., for molecular dynamics simulations of large molecules. Moreover, note that all ACSF and wACSF models presented here use the empirical scheme introduced in ref 24. Their performance can be improved by careful fine-tuning of the descriptors. However, such a procedure is typically tedious, especially considering the excellent out of the box performance of SchNet.

An interesting effect can be observed when comparing the performance of standard ACSFs to the recently suggested

wACSFs. In tasks that focus on modeling structurally and chemically diverse datasets (QM9), wACSF produce better results. However, in problems for which small variations of the molecular structure must be resolved (MD17), ACSFs outperform wACSFs. The reason for this behavior is the loss of spatial resolution of wACSF, which is a direct consequence of the improved elemental resolution. Whether this problem can be circumvented by learning elemental weights in a similar manner as in SchNet will be the focus of future research.

SchNet achieves chemically accurate prediction on the ANI-1 dataset. The ANI-1 neural network potential,[23] which is based on Behler−Parrinello networks, reported a RMSE of 1.2 kcal mol^{-1}, using 80% of the ANI-1 dataset for training and 10% for validation. Using SchNet, we already obtain a RMSE of 0.89 kcal mol^{-1}, using a training set of 10 million reference examples. Raising our splits up to 80% of the entire dataset for training and 10% for validation and testing, we obtain a MAE of 0.47 kcal mol^{-1} and a RMSE of 0.77 kcal mol^{-1}.

8. CONCLUSIONS

SchNetPack is a framework for neural networks of atomistic systems that simplifies accessing standard benchmark datasets, training models of different architectures, and evaluating their performance. It provides an interface to combine it with the functionality of the ASE package such as molecular dynamics simulations. We plan on extending SchNetPack further in the future by adding more datasets, advanced training mechanisms such as active sampling, support for additional quantum-mechanical observables, and further neural network architectures. We expect this unification and simplification to be of great value for the community, because it allows one to concentrate on the design of the neural network models and easily compare different architectures.

■ ASSOCIATED CONTENT

ⓢ Supporting Information

The Supporting Information is available free of charge on the ACS Publications website at DOI: 10.1021/acs.jctc.8b00908.

Details on the setup of the different experiments (PDF)

■ AUTHOR INFORMATION

Corresponding Authors
*E-mail: kristof.schuett@tu-berlin.de (K. T. Schütt).
*E-mail: alexandre.tkatchenko@uni.lu (A. Tkatchenko).
*E-mail: klaus-robert.mueller@tu-berlin.de (K.-R. Müller).

ORCID ⓘ
K. T. Schütt: 0000-0001-8342-0964
A. Tkatchenko: 0000-0002-1012-4854

Notes
The authors declare no competing financial interest.

■ REFERENCES

(1) Bartók, A. P.; Payne, M. C.; Kondor, R.; Csányi, G. Gaussian approximation potentials: The accuracy of quantum mechanics, without the electrons. *Phys. Rev. Lett.* **2010**, *104*, 136403.

(2) Rupp, M.; Tkatchenko, A.; Müller, K.-R.; Von Lilienfeld, O. A. Fast and accurate modeling of molecular atomization energies with machine learning. *Phys. Rev. Lett.* **2012**, *108*, 058301.

(3) Schütt, K. T.; Glawe, H.; Brockherde, F.; Sanna, A.; Müller, K.-R.; Gross, E. How to represent crystal structures for machine learning: Towards fast prediction of electronic properties. *Phys. Rev. B: Condens. Matter Mater. Phys.* **2014**, *89*, 205118.

(4) Behler, J. Constructing high-dimensional neural network potentials: A tutorial review. *Int. J. Quantum Chem.* **2015**, *115*, 1032−1050.

(5) Huo, H.; Rupp, M. Unified representation for machine learning of molecules and crystals. ArXiv preprint, *arXiv:1704.06439*, **2017**.

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

(7) De, S.; Bartók, A. P.; Csányi, G.; Ceriotti, M. Comparing molecules and solids across structural and alchemical space. *Phys. Chem. Chem. Phys.* **2016**, *18*, 13754−13769.

(8) Morawietz, T.; Singraber, A.; Dellago, C.; Behler, J. How van der Waals interactions determine the unique properties of water. *Proc. Natl. Acad. Sci. U. S. A.* **2016**, *113*, 8368−8373.

(9) Gastegger, M.; Behler, J.; Marquetand, P. Machine learning molecular dynamics for the simulation of infrared spectra. *Chem. Sci.* **2017**, *8*, 6924−6935.

(10) Chmiela, S.; Tkatchenko, A.; Sauceda, H. E.; Poltavsky, I.; Schütt, K. T.; Müller, K.-R. Machine learning of accurate energy-conserving molecular force fields. *Sci. Adv.* **2017**, *3*, No. e1603015.

(11) Faber, F. A.; Hutchison, L.; Huang, B.; Gilmer, J.; Schoenholz, S. S.; Dahl, G. E.; Vinyals, O.; Kearnes, S.; Riley, P. F.; von Lilienfeld, O. A. Prediction errors of molecular machine learning models lower than hybrid DFT error. *J. Chem. Theory Comput.* **2017**, *13*, 5255−5264.

(12) Podryabinkin, E. V.; Shapeev, A. V. Active learning of linearly parametrized interatomic potentials. *Comput. Mater. Sci.* **2017**, *140*, 171−180.

(13) Brockherde, F.; Vogt, L.; Li, L.; Tuckerman, M. E.; Burke, K.; Müller, K.-R. Bypassing the Kohn-Sham equations with machine learning. *Nat. Commun.* **2017**, *8*, 872.

(14) Bartók, A. P.; De, S.; Poelking, C.; Bernstein, N.; Kermode, J. R.; Csányi, G.; Ceriotti, M. Machine learning unifies the modeling of materials and molecules. *Sci. Adv.* **2017**, *3*, No. e1701816.

(15) Schütt, K. T.; Sauceda, H. E.; Kindermans, P.-J.; Tkatchenko, A.; Müller, K.-R. SchNet−a deep learning architecture for molecules and materials. *J. Chem. Phys.* **2018**, *148*, 241722.

(16) Chmiela, S.; Sauceda, H. E.; Müller, K.-R.; Tkatchenko, A. Towards exact molecular dynamics simulations with machine-learned force fields. *Nat. Commun.* **2018**, *9*, 3887.

(17) Ziletti, A.; Kumar, D.; Scheffler, M.; Ghiringhelli, L. M. Insightful classification of crystal structures using deep learning. *Nat. Commun.* **2018**, *9*, 2775.

(18) Dragoni, D.; Daff, T. D.; Csányi, G.; Marzari, N. Achieving DFT accuracy with a machine-learning interatomic potential: Thermomechanics and defects in bcc ferromagnetic iron. *Physical Review Materials* **2018**, *2*, 013808.

(19) Behler, J.; Parrinello, M. Generalized neural-network representation of high-dimensional potential-energy surfaces. *Phys. Rev. Lett.* **2007**, *98*, 146401.

(20) Montavon, G.; Hansen, K.; Fazli, S.; Rupp, M.; Biegler, F.; Ziehe, A.; Tkatchenko, A.; Lilienfeld, A. V.; Müller, K.-R. Learning

invariant representations of molecules for atomization energy prediction. In *Advances in Neural Information Processing Systems 25 (NIPS 2012)*, Lake Tahoe, NV, USA, Dec. 3−6,2012; pp 440−448.

(21) Montavon, G.; Rupp, M.; Gobre, V.; Vazquez-Mayagoitia, A.; Hansen, K.; Tkatchenko, A.; Müller, K.-R.; von Lilienfeld, O. A. Machine learning of molecular electronic properties in chemical compound space. *New J. Phys.* **2013**, *15*, 095003.

(22) Zhang, L.; Han, J.; Wang, H.; Car, R.; E, W. Deep potential molecular dynamics: a scalable model with the accuracy of quantum mechanics. *Phys. Rev. Lett.* **2018**, *120*, 143001.

(23) Smith, J. S.; Isayev, O.; Roitberg, A. E. ANI-1: an extensible neural network potential with DFT accuracy at force field computational cost. *Chem. Sci.* **2017**, *8*, 3192−3203.

(24) Gastegger, M.; Schwiedrzik, L.; Bittermann, M.; Berzsenyi, F.; Marquetand, P. wACSF − Weighted atom-centered symmetry functions as descriptors in machine learning potentials. *J. Chem. Phys.* **2018**, *148*, 241709.

(25) Schütt, K. T.; Arbabzadah, F.; Chmiela, S.; Müller, K. R.; Tkatchenko, A. Quantum-chemical insights from deep tensor neural networks. *Nat. Commun.* **2017**, *8*, 13890.

(26) Gilmer, J.; Schoenholz, S. S.; Riley, P. F.; Vinyals, O.; Dahl, G. E. Neural message passing for quantum chemistry. ArXiv preprint, *arXiv:1704.01212* **2017**.

(27) Schütt, K. T.; Kindermans, P.-J.; Sauceda, H. E.; Chmiela, S.; Tkatchenko, A.; Müller, K.-R. SchNet: A continuous-filter convolutional neural network for modeling quantum interactions. *Adv. Neural Inf. Process. Syst.* **2017**, 991−1001.

(28) Lubbers, N.; Smith, J. S.; Barros, K. Hierarchical modeling of molecular energies using a deep neural network. *J. Chem. Phys.* **2018**, *148*, 241715.

(29) Behler, J. Atom-centered symmetry functions for constructing high-dimensional neural network potentials. *J. Chem. Phys.* **2011**, *134*, 074106.

(30) Behler, J. First principles neural network potentials for reactive simulations of large molecular and condensed systems. *Angew. Chem., Int. Ed.* **2017**, *56*, 12828−12840.

(31) He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.* **2016**, 770−778.

(32) Larsen, A. H.; Mortensen, J. J.; Blomqvist, J.; Castelli, I. E.; Christensen, R.; Dulak, M.; Friis, J.; Groves, M. N.; Hammer, B.; Hargus, C.; Hermes, E. D.; Jennings, P. C.; Jensen, P. B.; Kermode, J.; Kitchin, J. R.; Kolsbjerg, E. L.; Kubal, J.; Kaasbjerg, K.; Lysgaard, S.; Maronsson, J. B.; Maxson, T.; Olsen, T.; Pastewka, L.; Peterson, A.; Rostgaard, C.; Schiotz, J.; Schutt, O.; Strange, M.; Thygesen, K. S.; Vegge, T.; Vilhelmsen, L.; Walter, M.; Zeng, Z.; Jacobsen, K. W. The atomic simulation environmenta Python library for working with atoms. *J. Phys.: Condens. Matter* **2017**, *29*, 273002.

(33) Ruddigkeit, L.; van Deursen, R.; Blum, L. C.; Reymond, J.-L. Enumeration of 166 Billion Organic Small Molecules in the Chemical Universe Database GDB-17. *J. Chem. Inf. Model.* **2012**, *52*, 2864−2875.

(34) Ramakrishnan, R.; Dral, P. O.; Rupp, M.; von Lilienfeld, O. A. Quantum chemistry structures and properties of 134 kilo molecules. *Sci. Data* **2014**, *1*, 140022.

(35) Smith, J. S.; Isayev, O.; Roitberg, A. E. ANI-1, A data set of 20 million calculated off-equilibrium conformations for organic molecules. *Sci. Data* **2017**, *4*, 170193.

(36) Jain, A.; Ong, S. P.; Hautier, G.; Chen, W.; Richards, W. D.; Dacek, S.; Cholia, S.; Gunter, D.; Skinner, D.; Ceder, G.; Persson, K. A. Commentary: The Materials Project: A materials genome approach to accelerating materials innovation. *APL Mater.* **2013**, *1*, 011002.

(37) Abadi, M.; Agarwal, A.; Barham, P.; Brevdo, E.; Chen, Z.; Citro, C.; Corrado, G. S.; Davis, A.; Dean, J.; Devin, M.; Ghemawat, S.; Goodfellow, I.; Harp, A.; Irving, G.; Isard, M.; Jia, Y.; Jozefowicz, R.; Kaiser, L.; Kudlur, M.; Levenberg, J.; Mané, D.; Monga, R.; Moore, S.; Murray, D.; Olah, C.; Schuster, M.; Shlens, J.; Steiner, B.; Sutskever, I.; Talwar, K.; Tucker, P.; Vanhoucke, V.; Vasudevan, V.; Viégas, F.; Vinyals, O.; Warden, P.; Wattenberg, M.; Wicke, M.; Yu, Y.; Zheng, X. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems.* 2015; available via the Internet at: https://www.tensorflow.org/.

(38) Paszke, A.; Gross, S.; Chintala, S.; Chanan, G.; Yang, E.; DeVito, Z.; Lin, Z.; Desmaison, A.; Antiga, L.; Lerer, A. Automatic differentiation in PyTorch. *NIPS 2017 Workshop Autodiff* **2017**.

(39) Jones, E.; Oliphant, T.; Peterson, P. *SciPy: Open Source Scientific Tools for Python.* 2001; available via the Internet at: http://www.scipy.org/.

(40) Huang, T.-W. *tensorboardX*; available via the Internet at: https://github.com/lanpa/tensorboardX.

(41) Collette, A. *Python and HDF5*; O'Reilly Media: Sebastopol, CA, USA, 2013.