Machine Learning: Science and Technology

PAPER • OPEN ACCESS

Training-free hyperparameter optimization of neural networks for electronic structures in matter

To cite this article: Lenz Fiedler *et al* 2022 *Mach. Learn.: Sci. Technol.* **3** 045008

View the article online for updates and enhancements.

You may also like

- Graph networks for molecular design
Rocío Mercado, Tobias Rastemo, Edvard Lindelöf et al.
- Efficient hyperparameter tuning for kernel ridge regression with Bayesian optimization
Annika Stuke, Patrick Rinke and Milica Todorovi
- Reinforcement learning enhanced quantum-inspired algorithm for combinatorial optimization
Dmitrii Beloborodov, A E Ulanov, Jakob N Foerster et al.

This content was downloaded from IP address 3.12.34.178 on 04/05/2024 at 21:29

IOP Publishing Mach. Learn.: Sci. Technol. 3 (2022) 045008 https://doi.org/10.1088/2632-2153/ac9956

MACHINE LEARNING
Science and Technology

OPEN ACCESS

RECEIVED
8 August 2022

REVISED
10 September 2022

ACCEPTED FOR PUBLICATION
11 October 2022

PUBLISHED
28 October 2022

Original Content from this work may be used under the terms of the Creative Commons Attribution 4.0 licence.

Any further distribution of this work must maintain attribution to the author(s) and the title of the work, journal citation and DOI.

PAPER

Training-free hyperparameter optimization of neural networks for electronic structures in matter

Lenz Fiedler[1,2,*], **Nils Hoffmann**[2], **Parvez Mohammed**[2], **Gabriel A Popoola**[3], **Tamar Yovell**[1], **Vladyslav Oles**[4], **J Austin Ellis**[4], **Sivasankaran Rajamanickam**[5] and **Attila Cangi**[1,*]

[1] Center for Advanced Systems Understanding (CASUS), Helmholtz-Zentrum Dresden-Rossendorf (HZDR), D-02826 Görlitz, Germany
[2] Technische Universität Dresden, D-01062 Dresden, Germany
[3] Elder Research, Inc., 300 West Main Street, Charlottesvile, VA 22903, United States of America
[4] Oak Ridge National Laboratory, Oak Ridge, TN 37830, United States of America
[5] Sandia National Laboratories, Albuquerque, NM 87185, United States of America
[*] Authors to whom any correspondence should be addressed.

E-mail: l.fiedler@hzdr.de and a.cangi@hzdr.de

Keywords: density functional theory, surrogate model, hyperparameter optimization, neural networks

Supplementary material for this article is available online

Abstract

A myriad of phenomena in materials science and chemistry rely on quantum-level simulations of the electronic structure in matter. While moving to larger length and time scales has been a pressing issue for decades, such large-scale electronic structure calculations are still challenging despite modern software approaches and advances in high-performance computing. The silver lining in this regard is the use of machine learning to accelerate electronic structure calculations—this line of research has recently gained growing attention. The grand challenge therein is finding a suitable machine-learning model during a process called hyperparameter optimization. This, however, causes a massive computational overhead in addition to that of data generation. We accelerate the construction of neural network models by roughly two orders of magnitude by circumventing excessive training during the hyperparameter optimization phase. We demonstrate our workflow for Kohn–Sham density functional theory, the most popular computational method in materials science and chemistry.

1. Introduction

The electronic structure of matter can be viewed as nature's glue [1] that binds atoms together into condensed systems like molecules and solids, thereby shaping the diversity of chemical systems and materials that surround us. A wide variety of materials characteristics including structural, elastic, and response properties are determined by the electronic structure [2]. Pressing questions from industry and society such as finding better materials for photovoltaics, identifying more efficient catalysts, designing future battery technologies, and discovering materials with novel properties are linked directly to the electronic structure of matter.

Electronic structure calculations are indispensable for complementing experimental investigations in materials science, and the need for ever more accurate and particularly efficient calculations is unbroken. Currently, the most widely used electronic structure method is Kohn–Sham density functional theory (DFT) [3–5] due to its balance of accuracy and computational efficiency. Under the assumption of the Born-Oppenheimer approximation [6], by employing the Kohn–Sham formalism, and the ever growing variety of exchange-correlation functionals [7], DFT enables electronic structure calculations for a large range of systems.

However, large-scale simulations at system sizes reaching millions of atoms, typically encountered in state-of-the-art molecular dynamics simulations, remain a final frontier for DFT. While DFT methods may possess favorable scaling properties compared to other *ab-initio* approaches, DFT calculations can usually

© 2022 The Author(s). Published by IOP Publishing Ltd

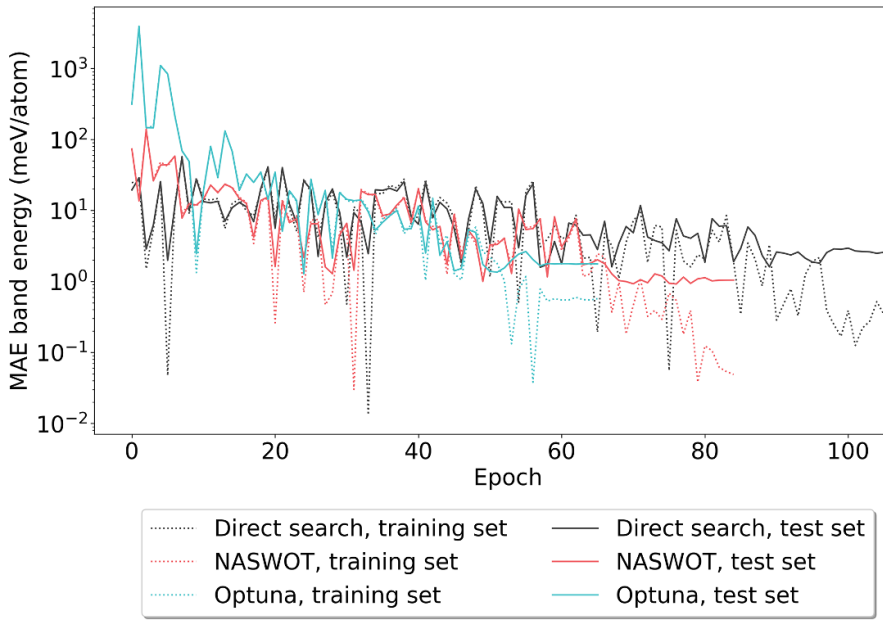

Figure 1. Training curves for three different model architectures and optimization parameters determined by three different hyperparameter optimization techniques. The loss in term of the physically motivated band energy on both the training (dashed) and test set (solid line) per epoch is shown. The grey curves correspond to direct search, the baseline for our manuscript as given in [35], the blue curves to the Optuna [37] library and the red curves to the NASWOT method [38]. Drastic differences can be observed, with NASWOT and Optuna outperforming the direct search approach in overall accuracy. Further, the Optuna model shows the best generalizability, as quantified by the smallest difference in test vs. training loss.

only be performed for hundreds and up to a few thousand atoms. This applies especially to dynamical settings [8] or to systems exposed to high temperatures [9]. Larger calculations can only be accomplished in special cases with enormous computational cost and time [10], thereby rendering large-scale investigations infeasible. The traditional pathway to circumventing these technical restrictions relies on algorithmic advances in software which leads to computationally more efficient calculations. Alternatively, approximate models such as average atom models [11–13] are applied for otherwise unattainable calculations. While they have a smaller computational overhead, they sacrifice accuracy.

However, a drastically different option is to combine the power of machine learning (ML) with DFT data. This emergent field of research is growing fast [14–18]. While ML approaches can be used to improve the accuracy and applicability of the DFT framework by constructing ML-based exchange-correlation functionals [19–21], the work presented here focuses on ML surrogate models that replace conventional DFT calculations. Two very popular ways to achieve this goal are the extraction of application-specific information from DFT data sets [22–24] and the construction of interatomic potentials [25–27] for molecular dynamics simulations based on Gaussian process regression [28], ridge regression [29], or neural networks (NNs) [30].

In this work, we focus on another emergent approach, namely the development of ML surrogate models that directly access the electronic structure without the need to perform DFT calculations. Pioneering efforts include kernel ridge-regression [31–33] and deep NN [33] models for predicting the electronic density. Based on these efforts, NN models for predicting the local density of states (LDOS) have recently been developed [34, 35]. These models are more general than those based solely on the electronic density. They replace traditional DFT calculations by enabling direct access to both the electronic structure and related observables, such as the total energy.

More specifically, we introduced such an LDOS-based framework in [35], where we show accurate NN models for aluminium at room temperature, as well as the melting point across liquid and solid configurations. In the course of developing such a framework, the lack of a general-purpose workflow for an automated generation of ML surrogate models became apparent, despite existing pioneering efforts [32]. The principle challenge we tackle in this work is to overcome the massive computational overhead due to hyperparameter optimization, which is a general problem for ML methodologies [36]. Determining suitable hyperparameters for a NN surrogate model governs model accuracy and generalizability, as is illustrated in figure 1 for different hyperparameter optimization techniques discussed in this manuscript. It constitutes a challenging task, even if the underlying electronic structure of the system is well understood (e.g. metals at room temperature, which we investigate in this work).

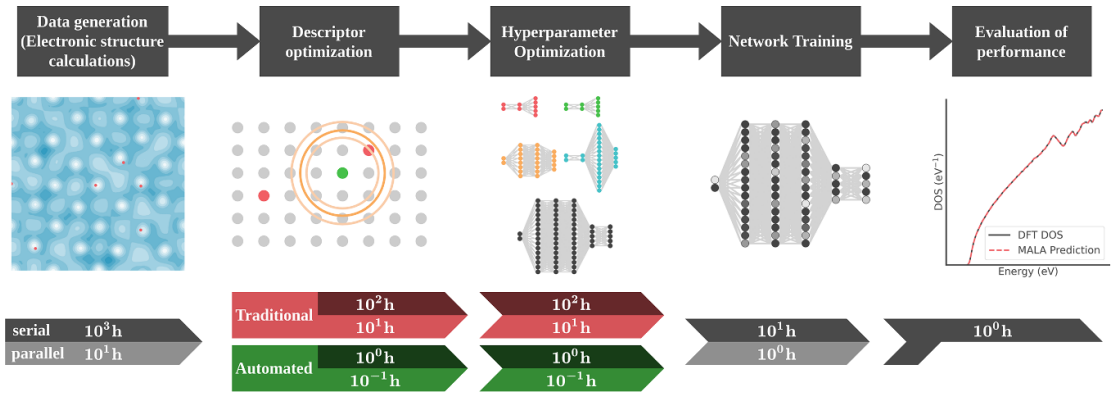

Figure 2. Schematic overview over the proposed workflow. Constructing surrogate models in the traditional fashion (red) is contrasted with our accelerated workflow (green). Here, h denotes CPU/GPU hours for serial execution or wall-time for parallel execution. Timings are estimated with justifications given in the supplementary note 3. The pictograms show from left to right: a contour plot of the electronic density of an aluminum cell for $z = 0$, with atoms close to the plane projected onto it in red; the descriptor calculation around a grid point (green dot) incorporating information of adjacent grid-points (grey) and atoms (red) within a certain radius (orange circles), which has to be optimized (light orange dots); different candidate networks; tuning of network weights; comparison of actual and predicted outputs (here, density of states).

A large amount of compute time must be dedicated to relatively long and inaccessible training and optimization processes, in which high-fidelity data sets have to be constructed, suitable hyperparameters have to be identified, and models have to be constructed. The major bottleneck in this process is not due to the computational overhead of data generation, as efficient DFT codes and automation frameworks [39–41] exist. While DFT calculations may require large amounts of computational power, they can be completed in a reasonable amount of *wall-time* by means of efficient parallelization and independent individual calculations. Instead, the principal time constraint for the creation of surrogate ML models lies in the hyperparameter optimization, which requires the repeated training of potentially suitable NNs, as illustrated in figure 2. The computational cost of NN training quickly becomes excessive, especially when considering a wide range of hyperparameters. While the training costs may be amortized by subsequent accelerated dynamical simulations, they render ML surrogate models prohibitive for many applications.

In this paper, we tackle this problem by providing a highly efficient and automated hyperparameter optimization workflow for generating ML surrogate models of electronic structures. It speeds up this process by two orders of magnitude (see figure 2) and comprises two central components—a *training-free score* and a *descriptor surrogate metric*. We adapt the recently developed technique on neural architecture search without training (NASWOT) [38] as a *training-free score* for electronic structures which does not require any NN training up until an optimal set of hyperparameters has been identified. It correlates well with the accuracy of a NN, as we demonstrate in a comprehensive comparison with state-of-the-art hyperparameter optimization techniques (see figure 3). Furthermore, we also introduce the average cosine similarity distance (ACSD) as a highly efficient *descriptor surrogate metric* for finding optimal descriptors for particle-mesh data (see figures 7 and 8).

We hence provide the basis for automated ML [42] workflows for modeling electronic structures. The resulting software framework, MALA [43], enables researchers to construct DFT surrogate models without extensive knowledge in ML or access to leadership-class computational infrastructure. We thus pave the way toward accessible and large-scale electronic structure calculations driven by ML.

2. Methods

2.1. Density functional theory

DFT, which we exclusively use in the Born-Oppenheimer approximation [6], is based upon the Hohenberg–Kohn theorems [3] and the Kohn–Sham formalism [4]. In the context of this work, we work within finite-temperature DFT [5], i.e. $\tau > 0\,K$. Within DFT, a system consisting of N_e electrons and N_i ions is treated. The ionic positions $\underline{\boldsymbol{R}}$. For ease of notation, we drop the parametric dependence on $\underline{\boldsymbol{R}}$ and energetic contributions due to ion–ion interactions in the following. Also note that here and below we adopt Hartree atomic units, i.e. $\hbar = e = m_e = a_0 = 1$.

The central quantity in DFT is the electronic density n, which is defined via the Kohn–Sham orbitals ϕ_j as

$$n(\boldsymbol{r}) = \sum_j f^\tau(\epsilon_j^\tau)\,|\phi_j(\boldsymbol{r})|^2\,, \tag{1}$$

with f^τ being the Fermi–Dirac distribution function and j running over all Kohn–Sham orbitals included in a particular DFT calculation. Their number has to be adjusted according to τ, as large temperatures lead to thermal excitations. The Kohn–Sham orbitals constitute a non-interacting auxiliary system restricted to reproduce the interacting electronic density and are governed by one-particle Schrödinger-like equations, often referred to as Kohn–Sham equations

$$\left[-\frac{1}{2}\nabla^2 + v_s^\tau(\boldsymbol{r})\right]\phi_j(\boldsymbol{r}) = \epsilon_j^\tau \phi_j(\boldsymbol{r}). \tag{2}$$

Here, $v_s^\tau(\boldsymbol{r})$ refers to the Kohn–Sham potential, which is determined self-consistently and incorporates a mean-field description of electron–electron interaction, exchange-correlation effects, and the electron-ion interaction, while ϵ_j^τ are the energy eigenvalues of $\phi_j(\boldsymbol{r})$. DFT calculations are performed in an iterative fashion, seeking to identify an electronic density which minimizes the total (free) energy

$$A_{\text{total}}^{\text{BO}}[n] = T_s[\phi_j] - k_B\tau S_s[\phi_j] + E_H[n] + E_{XC}^\tau[n] + E^{ei}[n]\,, \tag{3}$$

with the kinetic energy T_s and entropy S_s of the non-interacting system, the electrostatic interaction terms of the electronic density with itself E_H and with the ions E^{ei}. All energetic contributions not included in these terms are absorbed into the exchange-correlation functional E_{XC}^τ, thus keeping the framework formally exact. Practical calculations are enabled by appropriate approximations for E_{XC}^τ, such as the LDA-PW91 [44] or the PBE functional [45] for $\tau = 0\,K$; the development of functionals for $\tau > 0\,K$ is an area of active research [46–48].

In the DFT surrogate models we construct, the central role of the density is replaced by the LDOS $d(\epsilon, \boldsymbol{r})$ (LDOS), defined via

$$d(\epsilon, \boldsymbol{r}) = \sum_j |\phi_j(\boldsymbol{r})|^2\, \delta(\epsilon - \epsilon_j^\tau). \tag{4}$$

For practical calculations, $\delta(\epsilon - \epsilon_j^\tau)$ has to be approximated numerically, and the way this approximation is performed is one of the hyperparameters of the DFT surrogate workflow. The advantage of using the LDOS is that it gives direct access to the total free energy of a system, as both the electronic density, as well as the electronic density of states can be given in terms of the LDOS as

$$D(\epsilon) = \sum_j \delta(\epsilon - \epsilon_j^\tau) = \int d\boldsymbol{r}\, d(\epsilon, \boldsymbol{r}), \tag{5}$$

$$n(\boldsymbol{r}) = \sum_j f_j^\tau |\phi_j(\boldsymbol{r})|^2 = \int d\epsilon\, f^\tau(\epsilon) d(\epsilon, \boldsymbol{r}), \tag{6}$$

and equation (3) can be expressed in terms of n and D. More precisely, D can be used to calculate the electronic entropy and band energy E_b and with these the total free energy is determined

$$\begin{aligned} A_{\text{total}}^{\text{BO}}[d] = {} & E_b[D[d]] - k_B\tau S_s[D[d]] - E_H[n[d]] \\ & + E_{XC}^\tau[n[d]] - \int d\boldsymbol{r}\, v_{XC}^\tau(\boldsymbol{r}) n[d](\boldsymbol{r}). \end{aligned} \tag{7}$$

2.2. NN based surrogate models

Drawing on equation (7), one can construct surrogate models for DFT simulations by finding a suitable way to approximate the LDOS, which at each point in (simulated) space $\boldsymbol{r}$ is a vector in the ε dimension. For this task, we employ a NN.

NNs, which we will refer to as M, are powerful regression models consisting of layers of so called neurons or perceptrons [49]. Each neuron performs a linear operation using weights $\mathbf{W}$ and biases $\boldsymbol{b}$ on provided inputs $\boldsymbol{x}$ and thereafter applies a non-linear activation function σ, yielding intermediate outputs $\boldsymbol{y}$ as

$$\boldsymbol{y} = \sigma(\mathbf{W}\boldsymbol{x} + \boldsymbol{b})\,, \tag{8}$$

which serve as input for subsequent neurons. In connecting multiple layers containing an arbitrary number of neurons, in principle any function can be approximated, as long as the NN is tuned on an appropriate amount of data in a process called training [50, 51]. Such a training process is usually performed using gradient methods in an iterative fashion, based on back-propagation [52]. For the training of a NN, the available data is divided into a training data set (to calculate the gradients), a validation data set (to monitor

NN accuracy during training), and a test data set (unseen during training and used to verify performance after training is completed). Each pass of the entire training data set through the network is labelled an *epoch*. NN performance is highly dependent on the correct choice of hyperparameters $\boldsymbol{\lambda}$ that characterize both architecture and training policy for a NN, such as the number and width of individual layers. We use feed-forward NNs, in which each neuron of a layer is connected to all neurons of subsequent layers. NNs were constructed using the PyTorch library [53] and trained on a single GPU.

We use NNs to build a *per-grid-point* model, and map a vector at each point in space to a corresponding LDOS vector. To this end we employ spectral neighborhood analysis potential (SNAP) [54–57] descriptors $B(\boldsymbol{r}, j)$ that are calculated as functionals of $\underline{\boldsymbol{R}}$ and encode local information around a grid-point. At each point $\boldsymbol{r}$, a feed-forward NN $M[\boldsymbol{\lambda}]$ then performs a mapping of the type

$$\tilde{d}(\epsilon, \boldsymbol{r}) = M(B(j, \boldsymbol{r}))[\boldsymbol{\lambda}], \tag{9}$$

that depends on a set of hyperparameters $\boldsymbol{\lambda}$. Thereafter the LDOS is post-processed to compute the desired output quantities, like energies, forces, etc analytically.

Within this work, all NNs have been trained with an algorithm consistent with [35]. For all networks, *early-stopping* was employed, i.e. training was halted once improvement of validation accuracy starts to stagnate. How long such stagnation has to be present for the algorithm to stop is one of the hyperparameters we tune with the algorithm outlined below. More information on technical details of network training can be found in supplementary note 1.

2.3. Hyperparameter optimization

Hyperparameter optimization aims to find a set of hyperparameters $\boldsymbol{\lambda}$, such as the number or width of layers in an NN, that minimizes a loss term L. Generally, this is an optimization problem where we need to minimize the loss

$$L = L\left(Y, \tilde{Y}(X)\left[\boldsymbol{\lambda}\right]\right), \tag{10}$$

where Y is the function we seek to model and $\tilde{Y}$ is its prediction based on input data X. Solving this complex optimization problem is a formidable task and forms a computational bottleneck in all ML applications [36]. For applications in the realm of electronic structures, we choose

$$L = L\left(d(\epsilon, \boldsymbol{r}), \tilde{d}(\epsilon, \boldsymbol{r}), N\right), \tag{11}$$

where d denotes the LDOS, N the total number of atomic snapshots, and $\tilde{d}$ is calculated via equation (9).

Multiple loss metrics are conceivable in equation (11). Since we are not interested in the LDOS itself, L is chosen to be the mean absolute error (MAE) of some quantity calculated via the LDOS. This quantity is chosen to be the total free energy A, and the loss metric becomes

$$L = \frac{1}{N}\sum_{i=1}^{N}\left|A\left[d_i\right] - A\left[M(B_i)[\boldsymbol{\lambda}]\right]\right|, \tag{12}$$

where d_i is the reference LDOS obtained from DFT for an atomic configuration labeled by the index i, while the NN aims to reproduce d_i using the SNAP descriptors B_i. During hyperparameter optimization, A is usually replaced by the band energy E_b due to better computational accessibility, and equation (12) is evaluated only at the very end.

Hyperparameter optimization is usually performed by choosing a *candidate* model sampled from the entirety of potential hyperparameters and then optimizing this model, which constitutes a *trial*. Various hyperparameter optimization techniques differ in the way they propose candidate models and refine potential hyperparameter guesses based on the information provided. Within this work, we investigate several state-of-the-art methods, namely

(a) **Direct search** approach [58, 59], as performed in [35],
(b) Tree-structured Parzen Estimator (TPE) [60] as implemented in the software library **Optuna** [37],
(c) **Optuna** coupled to a **NASWOT-based pruner**,
(d) Orthogonal Array Tuning **OAT** [61] and
(e) **NASWOT** [38], both with **fixed** and **optimized** training schemes.

The direct search approach serves as a baseline that we have used in prior work. Therein, one optimizes one hyperparameter at a time while holding the others fixed and training multiple candidate networks per hyperparameter, thus progressively generating more suitable models. In a similar fashion, the TPE algorithm implemented in Optuna improves model performance progressively by constructing a model to determine the relationship between the hyperparameter values and the performance observed in trials. To increase robustness, we perform each trial multiple times and report an averaged result to Optuna, so as not to trap the algorithm in local minima due to unfavorable network initializations. Optuna can be coupled to a *pruner*, i.e. a metric that discards unpromising trials. As we demonstrate below, NASWOT can be employed in such a fashion.

NASWOT and OAT are in principle exhaustive and sample the entire search space presented. In the case of OAT this is done via constructing orthogonal arrays, a concept often employed in experimental design. The orthogonal arrays yield a compact list of potential hyperparameter combinations, that in principle covers the search space at drastically reduced computational cost. After measuring the performance of all trials in such a list, a range analysis is performed to extract the optimal set of hyperparameters from it. One important limitation of this approach is that orthogonal arrays do not exist for any arbitrary number of (hyper) parameters [62], thus one has to either restrict or artificially inflate the search space.

NASWOT achieves a full sampling of the search space by measuring the performance of individual candidate NNs without training the NN at all, thus allowing for a large number of trials to be performed. One can simply test all possible combinations of hyperparameters. The NASWOT method measures the ability of an NN to assign distinct predictions to distinct input data points upon initialization. It operates on the underlying assumption that a network that is able to distinguish well between different data points at initialization should be able to perform well after training. The performance before training is quantified via the Jacobian **J**, which is defined as the derivative of the predicted LDOS w.r.t. the SNAP descriptors. One then assigns a score to a given NN upon initialization defined as

$$S_{\mathrm{NASWOT}} = \sum_{i}^{N_{\mathrm{batch}}} \left[\log(\sigma_{J,i} + k) + (\sigma_{J,i} + k)^{-1}\right]. \tag{13}$$

The goal of this equation is to assign a score to each candidate network that is indicative of the performance of said network *after* training, i.e. without having to train the network at all. To this end, the correlation matrix of the Jacobian is calculated, and $\sigma_{J,i}$ are the eigenvalues of this correlation matrix. Low correlation within the Jacobian leads to higher scores in equation (13). The correlation matrix is calculated in equation (13) for N_{batch} samples passed through the NN via equation (9) and by drawing on k as a small parameter which ensures numerical stability. This methodology is introduced and discussed in detail in [38]. We adapt the calculation of these scores by calculating the NASWOT mean score across five network initializations in terms of equation (13) as

$$S'_{\mathrm{NASWOT}} = \bar{S}^{T}_{\mathrm{NASWOT}} + \sigma\left(S^{T}_{\mathrm{NASWOT}}\right), \tag{14}$$

where $\bar{S}^{T}_{\mathrm{NASWOT}}$ denotes the mean and $\sigma\left(S^{T}_{\mathrm{NASWOT}}\right)$ the standard deviation across the T individual scores. This is done to increase the robustness of S^{T}_{NASWOT} w.r.t. network initialization. Assuming S'_{NASWOT} is sufficiently correlated with the prediction accuracy of a NN after training, equation (14) provides a computationally inexpensive means for performing hyperparameter optimization. It replaces the usual loss metric, such as in equation (12), which is computationally heavy, because it needs to be computed after training. It is to note that while the loss in our framework needs to be minimized, the NASWOT score needs to be maximized.

However, NASWOT can only optimize the architecture of a NN. All hyperparameters related to model optimization cannot be treated by NASWOT, as no model is optimized. We therefore test NASWOT in two settings—one we deemed **fixed** (i.e. optimization parameters are set to fixed values) and one called **optimized**, where after having determined the optimal network architecture via NASWOT, optimization related hyperparameters are identified via a reduced Optuna study. Table 1 gives an overview over the hyperparameter optimization techniques discussed here. Computational details such as employed energy cutoffs, ***k***-grids and exchange-correlation functionals for the DFT simulations can be obtained via these references. Code used to conduct hyperparameter optimization as well as the relevant models can be found in [64].

Table 1. Overview over different hyperparameter techniques employed throughout this work, the hyperparameters they are capable of optimizing, stopping criteria, and parallelization capabilities.

Method	Optimized hyperparameters	Parallelization
Direct search	Architecture and Optimization.	Upper limit for parallelization is number of potential values per hyperparameter.
Optuna (Tree-Structured Parzen Estimator (TPE)	Architecture and Optimization.	Parallelization in principle unlimited, in practice too many parallel instances lead to more uninformed trials and slow convergence.
Optuna coupled to a NASWOT-based pruner	As above.	As above.
Orthogonal array tuning (OAT)	Architecture and Optimization.	Upper limit for the parallelization is the number of rows in the orthogonal array.
NASWOT (fixed)	Architecture.	Upper limit for the parallelization is the number of trials.
NASWOT (optimized)	Architecture and Optimization.	Architecture: as NASWOT (fixed); Optimization: as Optuna.

3. Results

Based on equation (11), we have carried out large-scale hyperparameter optimizations in order to identify the most suitable techniques for automated DFT surrogate model generation. Our results below are divided into two categories—(1) those for optimizing hyperparameters determining the NN architecture and (2) methods for choosing the most suitable descriptors. The former technique can be applied to any ML workflow which deals with a mapping of vector quantities, while the latter highlights how physical insight can be used to accelerate modeling specifically in the materials science domain.

3.1. Training-free hyperparameter optimization score

We compare the NASWOT mean score with state-of-the-art hyperparameter optimization methods and highlight its utility as a superior alternative to conventional hyperparameter optimization schemes that are based training. These optimizations share the common goal of identifying a NN architecture and training routine with a minimal prediction error in the shortest amount of time. Ideally, the accuracy of a NN should be independent of the NN initialization, while the inference time should be minimal.

In our assessment of hyperparameter optimization techniques, we consider a simulation cell containing 256 aluminum atoms at room temperature (298 K) and ambient mass density (2.699 g cc^{-1}) [65]. This system represents the complexity of learning electronic structure data while still being computationally tractable for extended studies [35].

To better quantify the accuracy of each hyperparameter optimization method, we train the model identified as optimal five times, each time using a different network initialization. Then, for each initialization, inference was performed across 10 atomic configurations that had previously been used in [35]. Using these prediction results, the MAE was calculated according to equation (12). In doing so, we get five MAEs, one for each network initialization per hyperparameter optimization method. We use this information to assess both accuracy and robustness of the listed hyperparameter optimization techniques. Details on the hyperparameter ranges used in these experiments are provided in table 1 of supplementary note 1.

Our central result is illustrated in figure 3. It shows the MAE vs. total compute time for a NN identified by the considered hyperparameter optimization techniques, while also quantifying the robustness of the NN by showing the spread for the MAE across different network initializations as antennas for the data points. It demonstrates that our *training-free score* implemented as the NASWOT mean score (red) provides a speed-up of two orders of magnitude while maintaining high accuracy comparable to the other methods. The average accuracy of the NASWOT NN is better then obtained from the direct search (black) and OAT (orange). Only the Optuna-based methods outperform it slightly, but at the price of a massive computational cost. Generally, it is quite evident that for the most part, increasing computational time yields more accurate NNs, and, as quantified by the standard deviation over the inference accuracy, more robust training routines. However, the pure Optuna approach (blue) identifies an NN with excellent inference and training performance at a cost almost two orders of magnitude higher then a direct search. The NASWOT mean score

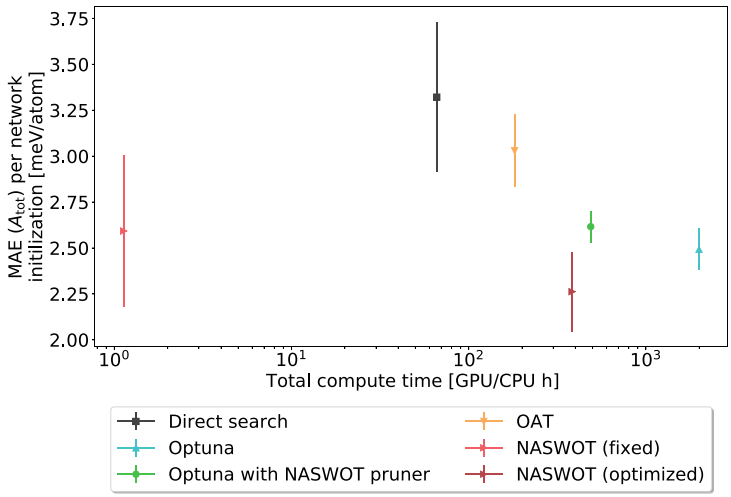

Figure 3. The *training-free score*, which we implemented as the NASWOT mean score (red), provides a speed up by two orders of magnitude compared to state-of-the-art hyperparameter optimization methods. We report the MAE across 10 atomic configurations vs. the computational demand. Markers indicate the average MAE and antennas the standard deviation over these MAE. The reported optimal NN architectures are obtained by completing five training cycles within each hyperparameter optimization method. The same methodology is employed in the subsequent figures.

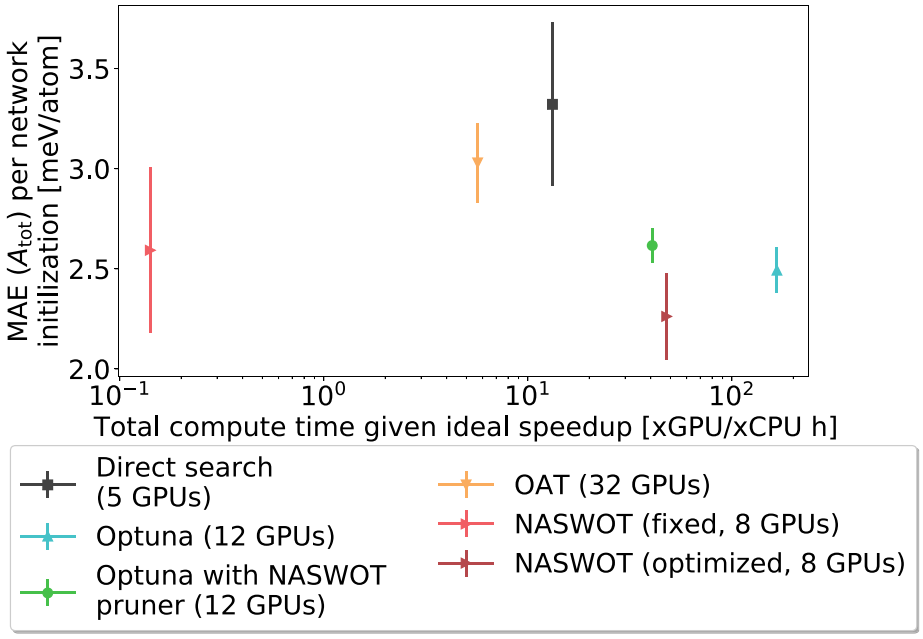

Figure 4. The *training-free score* (red) also provides a speed-up by two orders of magnitude when the time to solution is considered. Reported MAE (across 10 atomic configurations) vs. computational demand for different hyperparameter optimization techniques discussed above, with maximum expected speed-ups; actual speed-ups may differ slightly. Note that there are upper limits for parallelization. If too many GPUs are used for the Optuna parallelization, trials will be too independent, leading to slower convergence. The upper limit for OAT is the number of trials, since these are completely independent of each other. However, only a serial version of OAT was implemented for now. For the direct search, one GPU can be assigned per choice. For NASWOT there is no principal upper limit, but the same number of eight GPUs as in the Optuna study was chosen.

yields a NN with relatively large variances with respect to NN initializations. These are explained by the fact that NASWOT itself has no means to adjust parameters such as the learning rate. Therefore a fixed choice is required. Performing a small Optuna study afterwards (brown) drastically reduces this variance, while at the same time introducing an additional computational overhead. The length of such an Optuna study varies depending on demands for accuracy and availability of computational resources. Chiefly, a huge reduction in compute is enabled by our *training-free score* if a larger variance between network initializations can be tolerated. Even more accurate NNs become attainable with subsequent and slightly larger Optuna studies.

Naturally, one is often not directly interested in the total core hours, which measure both time and resources, but rather in the total time-to-result. To this end, figure 4 assesses speed-ups due to parallelization

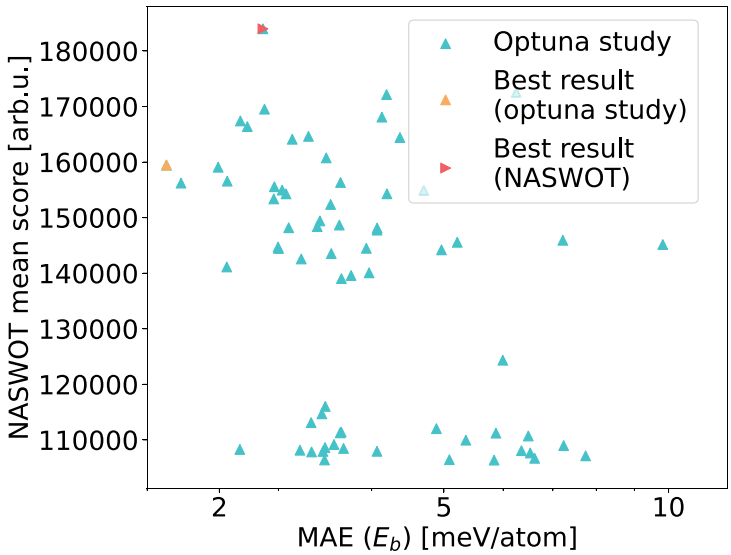

Figure 5. Training-free score (NASWOT mean score) vs. band energy prediction on the validation set of the Optuna study. The optimial NN architecture identified by NASWOT and Optuna are marked in red and orange, respectively.

of the hyperparameter studies. While this improves the performance across all hyperparameter optimization techniques, no drastic changes in their order can be observed. OAT has a more favorable scaling behavior then the direct search algorithm leading to a smaller runtime. Using Optuna to optimize training hyperparameters of the NASWOT NN yields the most accurate network in runtime of two days. Yet the principal result remains the same as in figure 3: NASWOT outperforms the other hyperparameter techniques again by two orders of magnitude. The central result displayed both in figures 3 and 4 is the first step toward automated ML surrogate model generation for electronic structures.

The NASWOT method relies on the correlation between the training-free score calculated upon initialization of an NN and the performance of said NN after training. Thus, the performance of NASWOT as shown in figures 3 and 4 itself is not sufficient to assess whether it is a reliable tool for automated surrogate model generation. Care has to be taken to ensure that such results are actually representative of a useful underlying correlation and not caused by, e.g. an imprecise problem statement. To this end, the correlation between the NASWOT mean score and the NN performance is analyzed in figure 5. The data basis for this figure is the Optuna study itself. Therefore training related hyperparameters can vary between different trial NNs. In doing so, it is ensured that the performance score assigned through this analysis is actually representative of suitable NN candidates. In order to save computation time, only the band energy rather than the total free energy was calculated for each candidate NN. The resulting comparison still provides the necessary insight, as it was shown in [35] that errors in the total free energy are dominated by errors in the band energy. It can be seen both visually and from the calculated Kendall τ coefficient [66] of -0.318 that the quantities shown in figure 5 are negatively correlated, meaning that large NASWOT scores relate to small NN errors, as is the expectation. While NASWOT and Optuna do not agree in their choice for the optimal NN architecture, the overall difference is not drastic; the optimal Optuna NN is still within the 20 best NNs according to the NASWOT score, while the NASWOT result yields a reasonable accuracy in the band energy. Based on these results, the NASWOT mean score is a suitable metric for identifying *a* NN capable of learning the LDOS to the desired accuracy.

Furthermore, the two clusters of points in figure 5 suggest that NASWOT can be used to optimize hyperparameter studies which are based on Optuna. A pruner can be constructed, that discards all candidate NNs below a certain threshold score. When treating new materials, such a threshold will be unknown beforehand and has to be estimated during runtime from preceding trials. Yet for first tests this simple implementation suffices. As shown in figure 4, the resulting hyperparameter optimization technique labeled as Optuna with NASWOT pruner (green) provides high accuracy and little uncertainty while coming at a significantly lower computational cost than the direct use of Optuna. However, one has to keep in mind that for unknown systems, additional computation time has to be included to accommodate for the incremental construction of the pruning threshold. Thus, such an approach is only a viable alternative to the NASWOT algorithm if computation time is not scarce, but yet not as abundant as necessary for a full Optuna study. Naturally, other pruning algorithms could be used to perform a similar task. OAT has been found to give an intermediate performance compared to the aforementioned methods. OAT is in principal highly parallel,

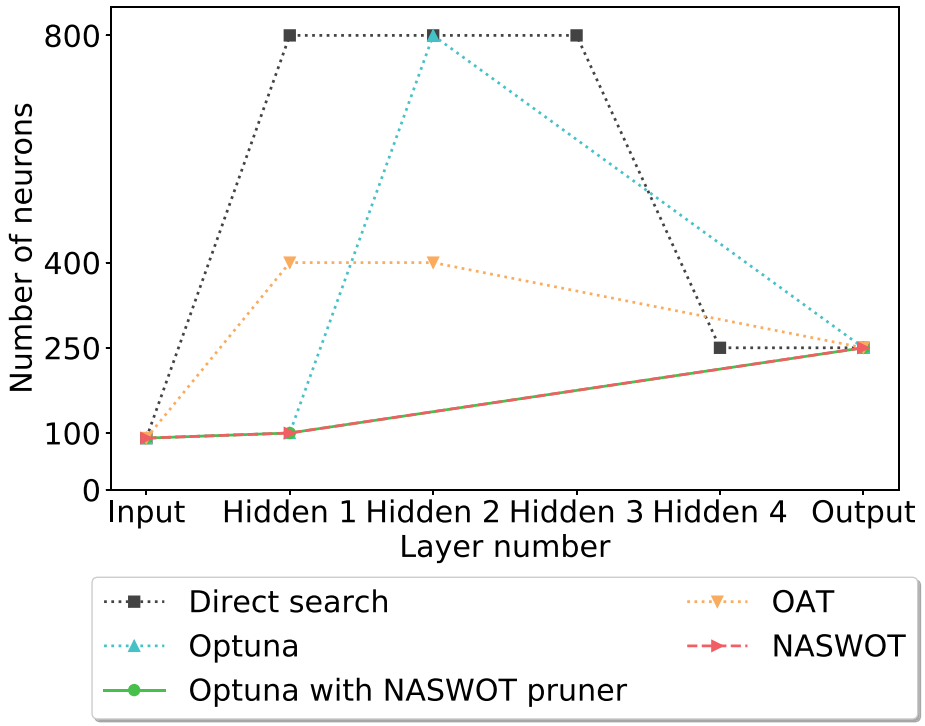

Figure 6. NN architectures identified as optimal by different hyperparameter optimizations.

giving it the best performance after NASWOT (provided enough GPUs are available), while at the same time outperforming a traditional direct search in terms of accuracy.

A final, important aspect of hyperparameter optimization is the identification of an *efficient* NN architecture. For initial studies, the size of a NN might not be crucial. But if surrogate models are to replace DFT calculations in dynamical simulations, then a minimal NN architecture resulting in minimal inference times would be desirable. To this end, figure 6 shows how the NNs identified by the different hyperparameter optimization methods differ greatly. The direct search favors a large NN, while the NASWOT mean score favors a small, single layer NNs. Optuna provides a middle ground, with a shallow NN, as does OAT. We deduce from figure 3 that the Optuna NN is close to the globally optimal NN architecture. The NASWOT mean score captures this optimum to a sufficient degree, while providing a massive reduction in computation time which is further illustrated in supplementary note 1. Overall, all of the NNs identified by the considered methods are smaller than those predicted by the direct search algorithm, resulting in drastically decreased inference times. As can be seen in figure 3, these smaller networks also yield higher accuracy. One explanation for this is a more favorable optimization landscape, or problems with over-fitting for large models.

3.2. Descriptor surrogate metric

While the aforementioned hyperparameter optimization is important to any conceivable ML problem that aims to map a vector quantity to another vector, there are, however, hyperparameters that are inherent to identifying suitable descriptors. In our grid-based approach to learning the electronic structure, we rely on descriptors that encode the local atomic environment around a point in the simulation cell. Since there is no clear physical relation between the NN prediction accuracy and the hyperparameters characterizing the way such a local environment is captured, data preprocessing itself requires a hyperparameter optimization. This requires the repeated training of a NN using a wide range of descriptors. The NASWOT mean score cannot be employed here, as it is not the network architecture we seek to optimize. We therefore introduce a *descriptor surrogate metric* called ACSD which is based on similarity measures. It facilitates identifying the optimal choice of descriptors for particle-mesh data. Similar to NASWOT, it is highly efficient and achieves a speed-up of two orders of magnitude compared to conventional NN training, because it enables a training-free optimization.

In our workflow, we employ the SNAP descriptors [54–57] to capture local atomic environments. However, local descriptors may be based on other established fingerprinting schemes for atomic configurations, such as SOAP [67], the Coulomb matrix [68], BoB [69], FCHL [70], or ACE [71, 72].

Assuming that we investigate cells consisting only of one chemical species, SNAP descriptors are calculated via the local atomic density around a grid-point

$$\rho(\boldsymbol{r}) = \delta(0) + \sum_{r_k < R_{\text{cut}}} f_c(r_k; R_{\text{cut}})\delta(\boldsymbol{r}_k), \tag{15}$$

after placing said grid-point at the origin. In equation (15), $\boldsymbol{r}_k$ with $r_k = |\boldsymbol{r}_k|$ is the position of atom k and R_{cut} is the cutoff radius which determines the length scale of the atomic environment considered by the SNAP descriptor. This local atomic density is represented in terms of four-dimensional hyperspherical coordinates. The number of terms in this expansion is denoted by j with a dimensionality governed by J_{max}; the higher J_{max}, the more components per grid-point are taken into account.

Our proposed surrogate metric is based on analyzing a similarity measure between the output and input vectors of the NN, namely the SNAP vectors which are the input and the LDOS vectors which are the output. Given two points on the real space grid of a simulation cell $\boldsymbol{r}_1$ and $\boldsymbol{r}_2$, we compute the cosine similarity S_C for either two LDOS vectors $d(\epsilon, \boldsymbol{r})$ and two SNAP vectors $B(j, \boldsymbol{r})$ as

$$S_C(X_1, X_2) = \frac{X_1 \cdot X_2}{\|X_1\|\|X_2\|} \tag{16}$$

with $X_1 = d(\epsilon, \boldsymbol{r}_1), X_2 = d(\epsilon, \boldsymbol{r}_2)$ or $X_1 = B(j, \boldsymbol{r}_1), X_2 = B(j, \boldsymbol{r}_2)$. We then compute a 2D point cloud $\left\{S_C^i(B), S_C^i(d)\right\}$ with $S_C^i(B) = S_C\{B(j, \boldsymbol{r}_t), B(j, \boldsymbol{r}_s)\}$, $S_C^i(d) = S_C\{d(\epsilon, \boldsymbol{r}_t), d(\epsilon, \boldsymbol{r}_s)\}$ for N_{sim} points $\{t, s\}$ sampled from the simulation cell. We consider $N_{\text{sim}} = 200 \times 200 = 40\,000$ points, i.e. for each grid point in a set of 200 grid points, these distances are determined w.r.t. 200 randomly drawn points.

The optimal choice of descriptors is determined by the hyperparameters R_{cut} and J_{max}. We need to consider two limiting cases: (i) when $S_C^i(B) \ll S_C^i(d)$, even drastically dissimilar descriptors might yield the same LDOS, and modeling becomes trivial from a ML perspective. This case applies when R_{cut} is small. However, this in turn means physically less informed descriptors and therefore lower prediction accuracy, because the length scale of the atomic environment is small. If we instead choose R_{cut} to be large, the descriptors will be physically well informed. But we risk approaching (ii) $S_C^i(B) \gg S_C^i(d)$, which makes our problem difficult to model in terms of ML. For a fixed choice of R_{cut}, the number of expansion coefficients governed by J_{max} determines which of these limiting cases is approached. As we aim for optimal performance w.r.t. both accuracy and data footprint, we argue that finding a combination of maximum R_{cut} and minimum J_{max} for which $S_C^i(B) \approx S_C^i(d)$ is the optimal choice.

To judge whether such a combination has been found, we introduce the ACSD. It is defined as the average difference

$$\text{ACSD} = \frac{1}{N_{\text{sim}}} \sum_{i}^{N_{\text{sim}}} |S_C^i(d) - S_C^i(B)| \tag{17}$$

between all points within the 2D point cloud $\left\{S_C^i(B), S_C^i(d)\right\}$ and $\left\{S_C^i(B), S_C^i(B)\right\}$, where N_{sim} denotes the number of sample points. The ACSD thus measures the average over the distribution of similarities that deviate from the $S_C^i(B) = S_C^i(d)$ line. Equation (17) can be evaluated rapidly in contrast to lengthy NN training required for a traditional hyperparameter search. To investigate the accuracy of the ACSD, we consider 20 snapshots of 128 beryllium atoms at room temperature (298 K) and ambient mass density ($1.896\ \text{g cc}^{-1}$) [63].

To assess the utility of the ACSD as a rapid and reliable *descriptor surrogate metric*, we compare the predicted ACSD with actual MAE of the total energy inferred from the NN. To that end we consider the hyperparameters $R_{\text{cut}} = 4.676$ Å and $J_{\text{max}} = 5$ which were identified as accurate in [35]. We then vary both hyperparameters, one at a time with the other held fixed. The reference result was calculated for all these hyperparameter combinations. This involved generating SNAP descriptors, training the NNs, and predicting the total free energy based on these NNs. The central results of this assessment are shown in figure 7 for a fixed number of components J_{max} and varying cutoff radius R_{cut}, and vice versa in figure 8.

It is evident in figure 7 that the NN prediction (green) becomes more accurate with increasing R_{cut}, up until a minimum at 4.676 Å (achieving the desired accuracy of below 3 meV atom^{-1}), after which a slight decrease in both average accuracy and spread is observed. This behavior is reproduced, almost exactly, by the ACSD surrogate metric (red). These results follow the intuitive expectation: a small R_{cut} means that individual SNAP descriptors carry less information about the atomic environment, making it harder for a NN to actually predict the electronic structure from the data provided. On the other hand, very large values of R_{cut} lead to SNAP descriptors that incorporate information from almost the entire simulation cell. These tend to be similar to one another, even though the actual electronic structure at a particular grid point

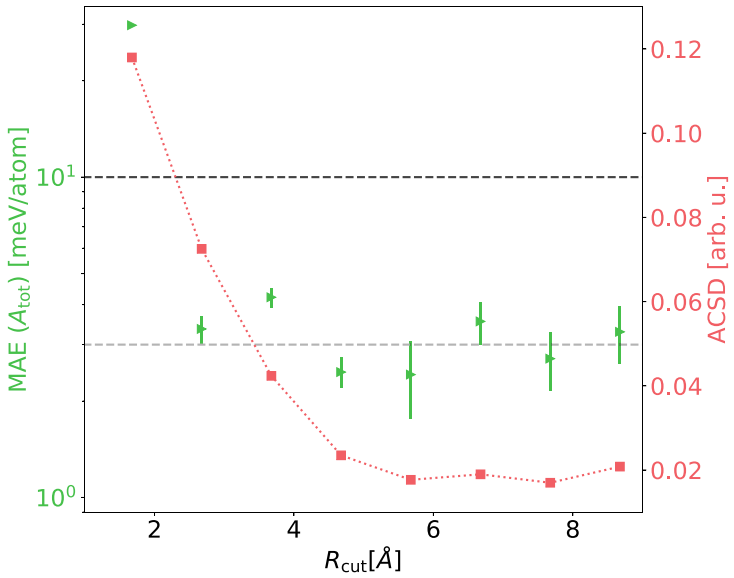

Figure 7. NN errors when using SNAP descriptors with differing R_{cut}. J_{max} was kept at 5. The dark dashed line indicates an error of 10 meV atom^{-1}, the upper threshold a model should achieve to be comparable to other competitive models, while the lighter dashed line indicates an error of 3 meV atom^{-1}, which is the accuracy reported in [35].

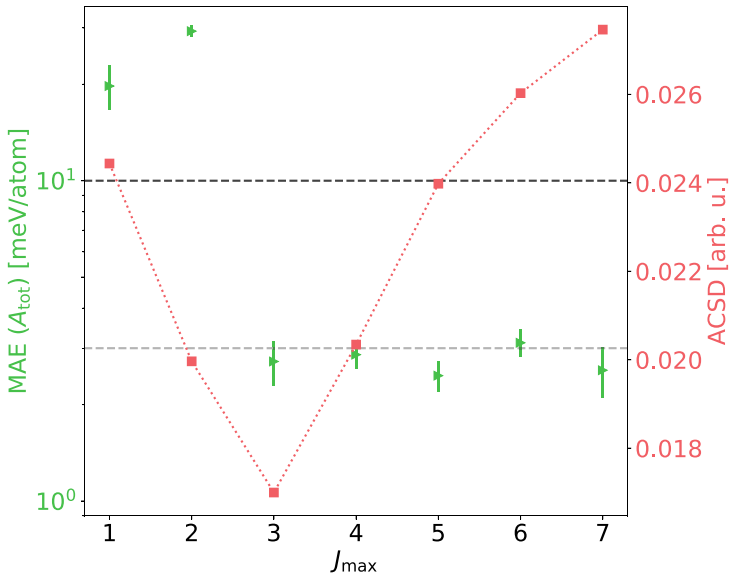

Figure 8. NN errors when using SNAP descriptors with differing J_{max}. R_{cut} was kept at 4.676 Å. The dark dashed line indicates an error of 10 meV atom^{-1}, the upper threshold a model should achieve to be comparable to other competitive models, while the lighter dashed line indicates an error of 3 meV atom^{-1}, which is the accuracy reported in [35].

differs, complicating training. Consequently, we expect the optimal R_{cut} in between these extremes, which is correctly identified by the ACSD. Furthermore, figure 7 confirms the prior assertion that ML modeling becomes increasingly more challenging as R_{cut} increases, as is evident from the increasing spread in the network prediction errors.

The computational speed-up of this method is drastic. The conventional method of finding optimal descriptors is a computationally heavy task, because it requires multiple NN model optimizations. Contrarily, the evaluation of the ACSD surrogate metric can be done in a matter of minutes on a single CPU. This results in a speed-up of around two orders of magnitude, while qualitatively yielding the same results.

The assessment for a fixed cutoff radius R_{cut} and varying J_{max} yields a similar trend which is illustrated in figure 8. As long as J_{max} is chosen sufficiently large, there is little effect on the accuracy of the NN (green), and above $J_{max} = 2$ all networks achieve the desired accuracy. This trend is not fully reflected by the ACSD surrogate metric (red). However, this can be explained due to the nature of this numerical experiment. An increase in components leads to additional components being added that are generally small in value. These

in turn cause slightly larger deviations for almost identical SNAP vectors, but do not carry meaningful information for the NN. This leads to almost unnoticeable differences in NN accuracies. However, minimal $J_{\max}$ values leading to reasonable accuracy is indeed reflected by the ACSD surrogate metric. Based only on the surrogate metric, one would choose the first data point for which the ACSD encounters a minimum, in this case $J_{\max} = 3$. The NN accuracy of this data set is comparable to that for higher $J_{\max}$ at reduced computational cost, thus demonstrating the utility of our ACSD surrogate metric.

4. Discussion and outlook

We tackle the complex optimization problem inherent to any NN surrogate model construction in the context of electronic structures. This constitutes a major computational bottleneck in addition to data generation. We provide a highly efficient and automated hyperparameter optimization workflow for generating ML surrogate models of electronic structures. It speeds up this process by two orders of magnitude (see figure 2), as we demonstrate in terms of a comprehensive comparison with state-of-the-art techniques. Our workflow consists of two advances—a *training-free score* for rapid hyperparameter optimization of NNs and a *descriptor surrogate metric* enabling an efficient search for suitable descriptors of particle-mesh data.

We have first assessed the accuracy and efficiency of our workflow against multiple hyperparameter optimization techniques such as the direct search, OAT, and the standard Optuna library. We achieve large gains in computational efficiency in hyperparameter optimization by adapting the NASWOT method [38] as a NASWOT mean score into our workflow. This method does not require any NN training up until an optimal set of hyperparameters has been identified. With our NASWOT mean score we are able to calculate a surrogate model in a few hours, whereas the state-of-the-art methods take days. In addition, if higher accuracy is needed, we showed that combining Optuna with the NASWOT mean score outperforms traditional search approaches. We also found that our hyperparameter optimization workflow impacts model performance. NN architectures identified as optimal by the NASWOT mean score algorithm are equally robust as the direct search, but yield smaller NNs with optimal inference performance.

Furthermore, we have developed the ACSD *descriptor surrogate metric* to find hyperparameters for the calculation of suitable particle-mesh descriptors without having to train any NN models. Likewise, our algorithm speeds up the state of the art by two orders of magnitude. First research into applying our models at larger temperatures suggest that the ACSD successfully recovers the expected physicys, i.e. favoring smaller cutoff radii (reflecting the larger disorder in the system).

By incorporating these two developments, we have devised a highly efficient ML surrogate modeling workflow shown in figure 2. All steps in this workflow can easily be automated with the algorithms considered here. These tools will enable a breadth of future applications in which large parts of the data processing for DFT surrogate models can be automated or executed with minimal user input. Our final workflow reduces the time required to construct surrogate models by two orders of magnitude. It thus provides a pathway to employing DFT surrogate models in large-scale investigations of materials under a variety of conditions.

Data availability statement

The data that support the findings of this study are openly available at the following URL/DOI: https://doi.org/10.14278/rodare.1107; https://doi.org/10.14278/rodare.1834; http://doi.org/10.23728/b2share.436e1e79daa54436a7703340431b4e19.

Acknowledgments

A C acknowledges useful discussions with Michael Bussmann. L F thanks Alexander Debus for providing us with additional computing time. We gratefully acknowledge computation time on the Bull Cluster at the Center for Information Services and High Performace Computing (ZIH) at Technische Universität Dresden. Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc. for the U.S. Department of Energy's National Nuclear Security Administration under Contract DE-NA0003525. This paper describes objective technical results and analysis. Any subjective views or opinions that might be expressed in the paper do not necessarily represent the views of the U.S. Department of Energy or the United States Government. This work was partially supported by the Center of Advanced Systems Understanding (CASUS) which is financed by Germany's Federal Ministry of Education and Research (BMBF) and by the Saxon state government out of the State budget approved by the Saxon State Parliament. This work was partially supported by the resources of the Oak Ridge Leadership Computing Facility, located in the National

Center for Computational Sciences at ORNL, which is managed by UT Battelle, LLC for the U.S. DOE (under the Contract No. DE- AC05-00OR22725).

ORCID iDs

Lenz Fiedler https://orcid.org/0000-0002-8311-0613
J Austin Ellis https://orcid.org/0000-0002-9901-102X
Sivasankaran Rajamanickam https://orcid.org/0000-0002-5854-409X
Attila Cangi https://orcid.org/0000-0001-9162-262X

References

[1] Kurth S and Perdew J P 2000 *Int. J. Quantum Chem.* **77** 814–8
[2] Martin R M 2004 *Electronic Structure: Basic Theory and Practical Methods* (Cambridge: Cambridge University Press)
[3] Hohenberg P and Kohn W 1964 *Phys. Rev.* **136** B864–71
[4] Kohn W and Sham L J 1965 *Phys. Rev.* **140** A1133–8
[5] Mermin N D 1965 *Phys. Rev.* **137** A1441–3
[6] Born M and Oppenheimer R 1927 *Ann. Phys., Lpz.* **389** 457–84
[7] Toulouse J 2021 arXiv:2103.02645
[8] Dziedzic J, Bhandari A, Anton L, Peng C, Womack J C Famili M, Kramer D and Skylaris C-K 2020 *J. Phys. Chem.* C **124** 7860–72
[9] Karasiev V V, Hinz J, Hu S X and Trickey S B 2021 *Nature* **600** E12–E14
[10] Nakata A *et al* 2020 *J. Chem. Phys.* **152** 164112
[11] Dharma-wardana M W C, Klug D D and Remsing R C 2020 *Phys. Rev. Lett.* **125** 075702
[12] Massacrier G, Böhme M, Vorberger J, Soubiran F and Militzer B 2021 *Phys. Rev. Res.* **3** 023026
[13] Callow T J, Kraisler E, Hansen S B and Cangi A 2021 arXiv:2103.09928
[14] Fiedler L, Shah K, Bussmann M and Cangi A 2021 arXiv:2110.00997
[15] Wei J, Chu X, Sun X Y, Xu K, Deng H X, Chen J Wei Z and Lei M 2019 *InfoMat* **1** 338–58
[16] Gubernatis J E and Lookman T 2018 *Phys. Rev. Mater.* **2** 120301
[17] Carleo G, Cirac I, Cranmer K, Daudet L, Schuld M, Tishby N Vogt-Maranto L and Zdeborová L 2019 *Rev. Mod. Phys.* **91** 045002
[18] Schmidt J, Marques M R G, Botti S and Marques M A L 2019 *npj Comput. Mater.* **5** 83
[19] Liu Q, Wang J, Du P, Hu L, Zheng X and Chen G 2017 *J. Phys. Chem.* A **121** 7273–81
[20] Schmidt J, Benavides-Riveros C L and Marques M A L 2019 *J. Phys. Chem. Lett.* **10** 6425–31
[21] Kirkpatrick J *et al* 2021 *Science* **374** 1385–9
[22] Ulissi Z W, Medford A J, Bligaard T and Nørskov J K 2017 *Nat. Commun.* **8** 14621
[23] Schütt K T, Glawe H, Brockherde F, Sanna A, Müller K R and Gross E K U 2014 *Phys. Rev.* B **89** 205118
[24] Ramakrishnan R, Dral P O, Rupp M and von Lilienfeld O A 2015 *J. Chem. Theory Comput.* **11** 2087–96
[25] Deringer V L and Csányi G 2017 *Phys. Rev.* B **95** 094203
[26] Sosso G C, Miceli G, Caravati S, Behler J and Bernasconi M 2012 *Phys. Rev.* B **85** 174103
[27] Morawietz T and Rg Behler J 2013 *J. Phys. Chem.* A **117** 32
[28] Bartók A P, Payne M C, Kondor R and Csányi G 2010 *Phys. Rev. Lett.* **104** 136403
[29] Schmidt J, Shi J, Borlido P, Chen L, Botti S and Marques M A L 2017 *Chem. Mater.* **29** 5090–103
[30] Smith J S, Isayev O and Roitberg A E 2017 *Chem. Sci.* **8** 3192–203
[31] Snyder J C, Rupp M, Hansen K, Müller K R and Burke K 2012 *Phys. Rev. Lett.* **108** 1079–7114
[32] Brockherde F, Vogt L, Li L, Tuckerman M E, Burke K and Müller K R 2017 *Nat. Commun.* **8** 872
[33] Tsubaki M and Mizoguchi T 2020 *Phys. Rev. Lett.* **125** 206401
[34] Chandrasekaran A, Kamal D, Batra R, Kim C Chen L and Ramprasad R 2019 *npj Comput. Mater.* **5** 22
[35] Ellis J A , Fiedler L, Popoola G A, Modine N A, Stephens J A, Thompson A P, Cangi A and Rajamanickam S 2021 *Phys. Rev.* B **104** 035120
[36] Hutter F, Lücke J and Schmidt-Thieme L 2015 *Künstl Intell.* **29** 329–37
[37] Akiba T, Sano S, Yanase T, Ohta T and Koyama M 2019 Optuna: a next-generation hyperparameter optimization framework *KDD '19: Proc. 25th ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining* (Association for Computing Machinery) pp 2623–31
[38] Mellor J, Turner J, Storkey A and Crowley E J 2020 arXiv:2006.04647v1
[39] Huber S P *et al* 2020 *Sci. Data* **7** 300
[40] Larsen A H *et al* 2017 *J. Phys.: Condens. Matter* **29** 273002
[41] Uhrin M, Huber S P, Yu J, Marzari N and Pizzi G 2021 *Comput. Mater. Sci.* **187** 110086
[42] Chauhan K, Jani S, Thakkar D, Dave R, Bhatia J and Tanwar S *et al* 2020 Automated machine learning: the new wave of machine learning *2020 Int. Conf. on Innovative Mechanisms for Industry Applications (ICIMIA) 2020—Proc.* pp 205–12
[43] Cangi A *et al* 2021 *Software publication* "*MALA*" (https://doi.org/10.5281/zenodo.5557254)
[44] Ceperley D M and Alder B J 1980 *Phys. Rev. Lett.* **45** 566–9
[45] Perdew J P, Burke K and Ernzerhof M 1996 *Phys. Rev. Lett.* **77** 3865
[46] Groth S, Dornheim T, Sjostrom T, Malone F D, Foulkes W M C and Bonitz M 2017 *Phys. Rev. Lett.* **119** 135001
[47] Karasiev V V, Sjostrom T, Dufty J and Trickey S B 2014 *Phys. Rev. Lett.* **112** 076403
[48] Brown E W, DuBois J L, Holzmann M and Ceperley D M 2013 *Phys. Rev.* B **88** 081102
[49] Rosenblatt F 1957 The perceptron: a perceiving and recognizing automaton (Project PARA) *Technical Report 85-460-1* (Cornell Aeronautical Laboratory)
[50] Hornik K, Stinchcombe M and White H 1989 *Neural Netw.* **2** 359–66
[51] Hornik K 1991 *Neural Netw.* **4** 251–7
[52] Goodfellow I, Bengio Y and Courville A 2016 *Deep Learning Book* illustrated edn (Cambridge, MA: The MIT Press)
[53] Paszke A *et al* 2019 PyTorch: an imperative style, high-performance deep learning library *Advances in Neural Information Processing Systems* vol 32 (Curran Associates, Inc.)
[54] Thompson A P, Swiler L P, Trott C R, Foiles S M and Tucker G J 2015 *J. Comput. Phys.* **285** 316–30

[55] Wood M A and Thompson A P 2018 *J. Chem. Phys.* **148** 241721
[56] Wood M A, Cusentino M A, Wirth B D and Thompson A P 2019 *Phys. Rev.* B **99** 184305
[57] Cusentino M A, Wood M A and Thompson A P 2020 *J. Phys. Chem.* A **124** 5456–64
[58] Hooke R and Jeeves T A 1961 *J. ACM* **8** 212–29
[59] Lewis R M, Torczon V and Trosset M W 2000 *J. Comput. Appl. Math.* **124** 191–207
[60] Bergstra J, Bardenet R, Bengio Y and Kégl B 2011 *Advances in Neural Information Processing Systems* vol 24
[61] Zhang X, Chen X, Yao L, Ge C and Dong M 2019 Deep neural network hyperparameter optimization with orthogonal array tuning *Neural Information Processing*, ed T Gedeon, K W Wong and M Lee (Cham: Springer) pp 287–95
[62] Beder J H and McComack M A 2017 *Commun. Stat. - Theory Methods* **46** 3690–7
[63] Fiedler L and Cangi A 2022 LDOS/SNAP data for MALA: Beryllium at 298 K (https://doi.org/10.14278/rodare.1834)
[64] Fiedler L, Hoffmann N, Mohammed P, Yovell T, Oles V, Ellis J A, Siva R and Attila C 2022 Scripts and networks for "Electronic structure machine learning surrogates without training" (https://doi.org/10.23728/b2share.436e1e79daa54436a7703340431b4e19)
[65] Ellis J A *et al* 2021 LDOS/SNAP data for MALA: aluminium at 298 K and 933 K (https://doi.org/10.14278/rodare.1107)
[66] Kendall M G 1938 *Biometrika* **30** 81–93
[67] Bartók A P, Kondor R and Csányi G 2013 *Phys. Rev.* B **87** 184115
[68] Rupp M, Tkatchenko A, Müller K R and Von Lilienfeld O A 2012 *Phys. Rev. Lett.* **108** 058301
[69] Hansen K , Biegler F, Ramakrishnan R, Pronobis W, von Lilienfeld O A, Müller K-R and Tkatchenko A 2015 *J. Phys. Chem. Lett.* **6** 1948–7185
[70] Faber F A, Christensen A S, Huang B and von Lilienfeld O A 2017 arXiv:1712.08417
[71] Drautz R 2019 *Phys. Rev.* B **99** 014104
[72] Lysogorskiy Y *et al* 2021 *npj Comput. Mater.* **7** 1–12