Finding Electronic Structure Machine Learning Surrogates without Training

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

[1] *Center for Advanced Systems Understanding (CASUS), D-02826 Görlitz, Germany*
[2] *Helmholtz-Zentrum Dresden-Rossendorf (HZDR), D-01328 Dresden, Germany*
[3] *Technische Universität Dresden, D-01062 Dresden, Germany*
[4] *Sandia National Laboratories, Albuquerque, NM 87185, USA*
[5] *Oak Ridge National Laboratory, Oak Ridge, TN 37830, USA*

(Dated: February 24, 2022)

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 machine-learning surrogate 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.

I. 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 only be performed for a few thousand atoms. This applies especially in 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].It currently focuses on extracting application-specific information from DFT data sets [19–21] and constructing interatomic potentials [22–24] for molecular dynamics simulations based on Gaussian process regression [25], ridge regression [26], or neural networks (NNs) [27].

Here, we however focus on using ML to directly tackle the electronic structure problem in terms of the Kohn-Sham equations [4]. Pioneering efforts include kernel ridge-regression [28–30] and deep NN [30] models for predicting the electronic density. Based on these efforts, NN models for predicting the local density of states (LDOS) have recently been developed [31, 32]. 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.

Despite these pioneering efforts [29], a general-purpose workflow for automated ML applications which can

* l.fiedler@hzdr.de
† a.cangi@hzdr.de

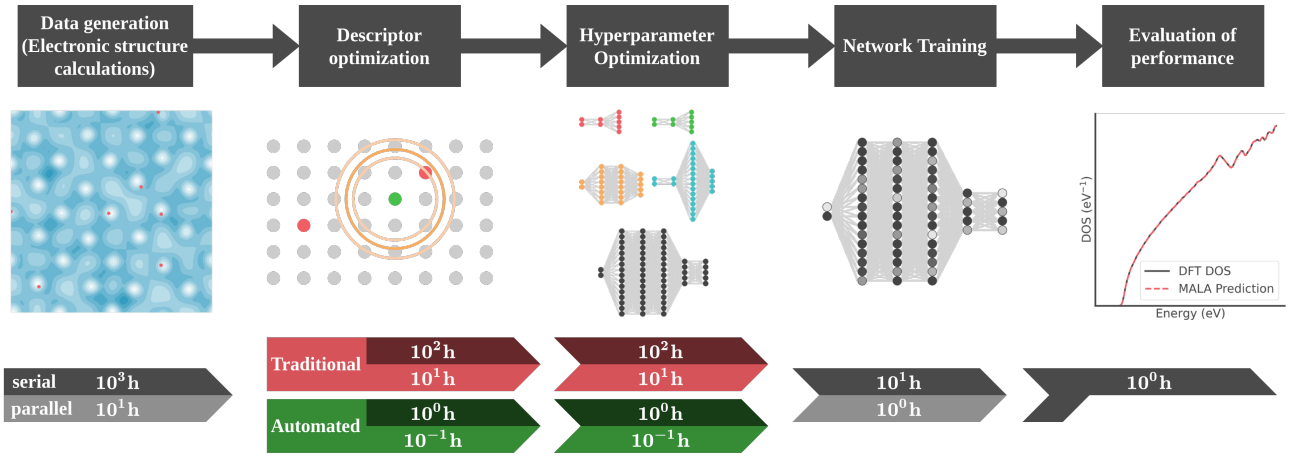

FIG. 1. Schematic overview over the proposed workflow. Constructing surrogate models in the traditional fashion (red) is contrasted with our accelerated workflow (green). Here, h denotes CPU/GPU hours for serial execution or wall-time for parallel execution. Timings are estimated with justifications 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 density of states.

tackle electronic structures has yet to come. The principle challenge is to overcome the massive computational overhead due to hyperparameter optimization. More specifically, 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 [33–35] 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 Fig. 1. 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 Fig. 1) 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) [36] 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 Fig. 2). 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 Fig. 6 and Fig. 7).

We hence provide the basis for automated ML [37] workflows for modeling electronic structures. The resulting software framework, MALA [38], enables researchers to construct DFT surrogate models without extensive knowledge in ML or access to leadership-class computational infrastructure. We thus pave the way towards accessible and large-scale electronic structure calculations driven by ML.

II. RESULTS

We begin by considering the task of any hyperparameter optimization, namely, finding 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 minimize the loss

$$L = L\left(Y, \tilde{Y}(X)[\boldsymbol{\lambda}]\right) \ , \tag{1}$$

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 [39].

For applications in the realm of electronic structures,

we choose

$$L = L\left(d(\epsilon, \boldsymbol{r}), \tilde{d}(\epsilon, \boldsymbol{r}), N\right) \ , \tag{2}$$

where d denotes the LDOS and N the total number of atomic snapshots. Our surrogate NN model $M[\boldsymbol{\lambda}]$ yields as output the predicted LDOS

$$\tilde{d}(\epsilon, \boldsymbol{r}) = M(B(j, \boldsymbol{r}))[\boldsymbol{\lambda}] \tag{3}$$

at each point in space $\boldsymbol{r}$ for a given descriptor $B(j, \boldsymbol{r})$. Each NN is trained until a stopping criterion is reached. Further details of our particular NN surrogate model are provided in the Supplementary Methods Section and in Ref. 32. We select the Spectral Neighborhood Analysis Potential (SNAP) [40–43] as a suitable descriptor as input to our NN model. It is calculated in terms of the atomic positions $\boldsymbol{R}$, i.e., $B(j, \boldsymbol{r}) = B(j, \boldsymbol{r})[\boldsymbol{R}]$ with components j. Note that $\boldsymbol{R}$ is a shorthand notation for the collection of the cartesian coordinates of all atoms. The SNAP descriptor provides a suitable representation of the local atomic environment around a grid-point $\boldsymbol{r}$ in the simulation cell.

Multiple loss metrics are conceivable in Eq. (2). 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

$$L = \frac{1}{N} \sum_{i=1}^{N} \left| A\big[d_i\big] - A\big[M(B_i)[\boldsymbol{\lambda}]\big] \right| , \tag{4}$$

where d_i is the reference LDOS obtained from DFT for an atomic configuration labelled by the index i, while the NN aims to reproduce d_i using the SNAP descriptors B_i. During hyperparameter optimization, the A functional is sometimes replaced by the band energy E_b functional due to better computational accessibility, and Eq. (4) is evaluated only at the very end.

Based on Eq. (3), 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 – (i) 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.

A. Training-free hyperparameter optimization score

We achieve training-free hyperparameter optimization in building NN surrogate models for electronic structure applications by implementing the NASWOT method [36] in our MALA framework. The NASWOT method relies on the correlation between the input and output of a NN. Here, this correlation is quantified in terms of a Jacobian $\mathbf{J}$ which is defined as the derivative of the predicted LDOS w.r.t. the SNAP descriptors. It assigns a score to a given NN upon initialization defined as

$$S_{\mathrm{NASWOT}} = \sum_{i}^{N_{\mathrm{batch}}} \left[\log(\sigma_{J,i} + k) + (\sigma_{J,i} + k)^{-1} \right] . \tag{5}$$

Here, N_{batch} denotes the size of a subset of the training data passed through the NN from Eq. (3), $\sigma_{J,i}$ the eigenvalues of the correlation matrix obtained from the Jacobian, and k a small parameter which ensures numerical stability. We then calculate the NASWOT mean score across five network initializations in terms of Eq. (5) as

$$S'_{\mathrm{NASWOT}} = \bar{S}^{T}_{\mathrm{NASWOT}} + \sigma\left(S^{T}_{\mathrm{NASWOT}}\right) \ , \tag{6}$$

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 robustness of S^{T}_{NASWOT} w.r.t. network initialization. S'_{NASWOT} then serves as a metric to determine how well an untrained NN can distinguish between given data points. The underlying assumption is that a network performing well at this task upon initialization will also yield a high accuracy after training. Further details on the implemented NASWOT mean score are provided in the Supplementary Methods Section. Assuming S'_{NASWOT} is sufficiently correlated with the prediction accuracy of a NN after training, Eq. (6) provides a computationally inexpensive means for performing hyperparameter optimization. It replaces the usual loss metric, such as in Eq. (4), which is computationally heavy, because it needs to be computed after training.

We compare NASWOT mean score with state-of-the-art hyperparameter optimization methods and highlight its utility as a superior alternative. 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. To that end, we compare the following hyperparameter optimization strategies:

1. **Direct search** approach [44, 45], as performed in Ref. [32],
2. Tree-structured Parzen Estimator (TPE) [46] as implemented in the software library **Optuna** [47],
3. **Optuna** coupled to a **NASWOT-based pruner**,
4. Orthogonal Array Tuning **OAT** [48] and
5. **NASWOT** [36], both with **fixed** and **optimized** training schemes.

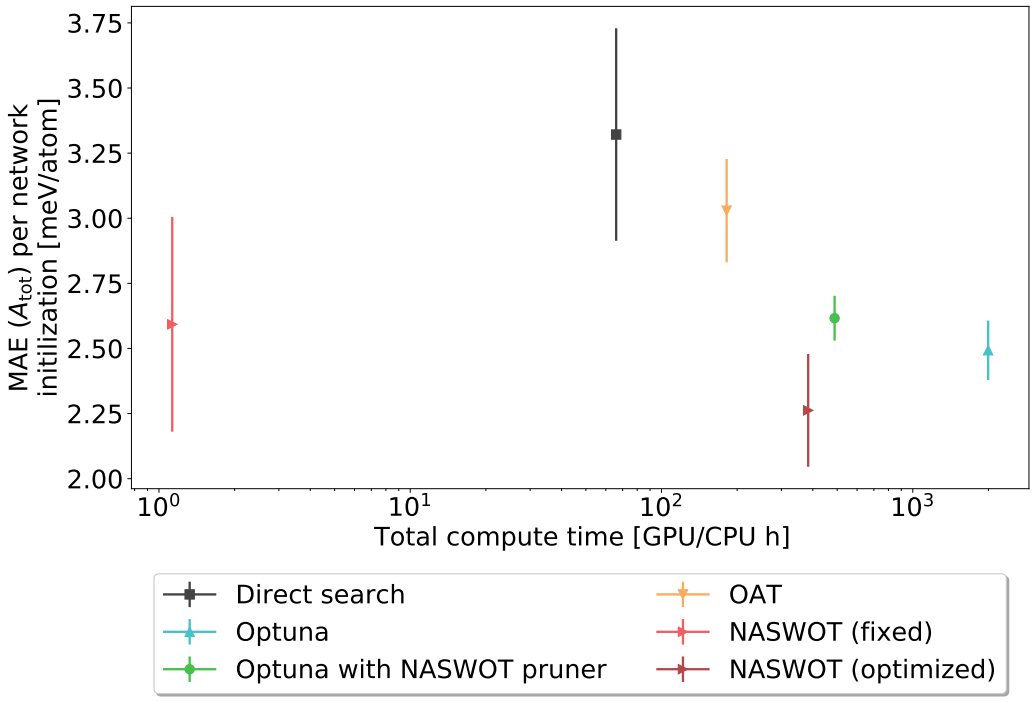

FIG. 2. The *training-free score*, which we implemented as the NASWOT mean score (red), provides a speed up by two orders of magnitude compared to state-of-the-art hyperparameter optimization methods. We report the mean absolute errors (MAE) across 10 atomic configurations vs. the computational demand. 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.

We provide further details of these algorithms in the Supplementary Methods Section.

In order to assess these 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) [49]. This system represents the complexity of learning electronic structure data while still being computationally tractable for extended studies [32].

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 available in Ref. [32]. Using these prediction results, the MAE was calculated according to Eq. (4). 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 Fig. 2. It shows the MAE vs. total compute time for a NN identified by the considered hyperparameter optimization techniques. 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 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 parameter 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 initializa-

tions 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, Fig. 3 assesses speed-ups due to parallelization 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 Fig. 2: NASWOT outperforms the other hyperparameter techniques again by two orders of magnitude. The central result displayed both in Fig. 2 and Fig. 3 is the first step towards automated ML surrogate model generation for electronic structures.

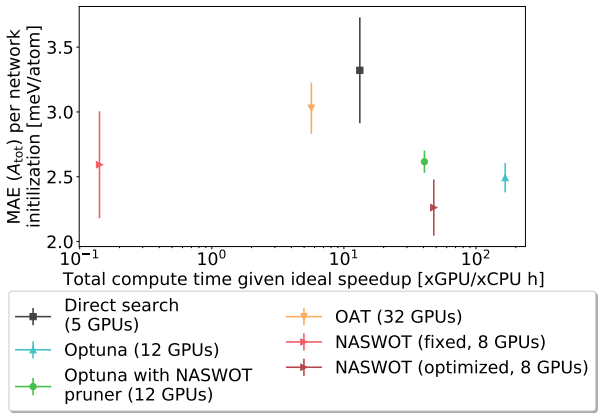

FIG. 3. The *training-free score* (red) also provides a speed-up by two orders of magnitude when the time to solution is considered. Reported MAE (across 10 atomic configurations) vs. computational demand for different hyperparameter optimization techniques discussed above, with maximum expected speed-ups; 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.

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 Fig. 2 and Fig. 3 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 Fig. 4. The data basis for this figure is the Optuna

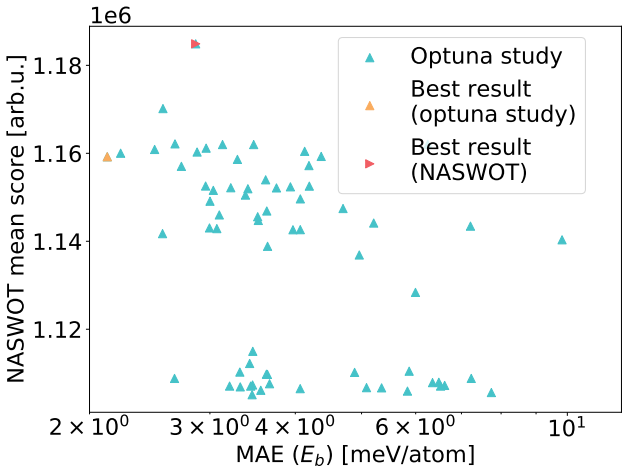

FIG. 4. Training-free score (NASWOT mean score) vs. band energy prediction on the validation set of the Optuna study. The optimial NN architecture identified by NASWOT and Optuna are marked in red and orange, respectively.

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 then the total free energy was calculated for each candidate NN. The resulting comparison still provides the necessary insight, as it was shown in Ref. [32] 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 [50] of -0.318 that the quantities shown in Fig. 4 are negatively correlated. 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 15 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 Fig. 4 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 Fig. 3, the resulting hyperparameter optimization technique labelled as Optuna with NAS-

WOT 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, giving it the best performance after NASWOT (provided enough GPUs are avaialble), 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 as 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, Fig. 5 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 Fig. 2 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.

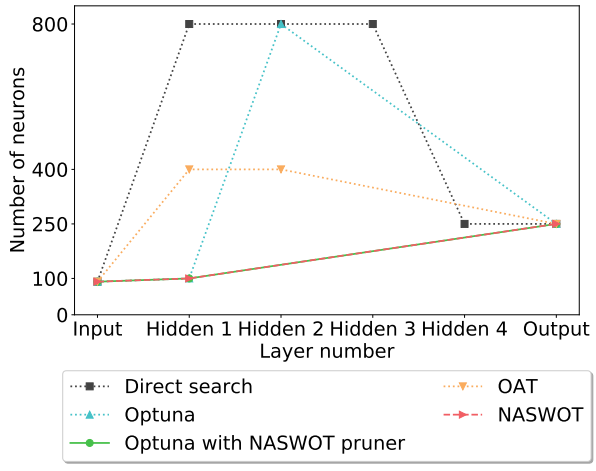

FIG. 5. NN architectures identified as optimal by different hyperparameter optimizations.

B. 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 [40–43] to capture local environments. However, local descriptors may be based on other established fingerprinting schemes for atomic configurations, such as SOAP [51], the Coulomb matrix [52], BoB [53], FCHL [54], or ACE [55, 56]. Assuming that we investigate cells consisting only of one chemical species, SNAP descriptors are calculated via the local atomic density around a grid-point

$$\rho(\boldsymbol{r}) = \delta(\boldsymbol{0}) + \sum_{r_k < R_{\mathrm{cut}}} f_c(r_k; R_{\mathrm{cut}})\delta(\boldsymbol{r}_k)\ , \tag{7}$$

after placing said grid-point at the origin. In Eq. (7), $\boldsymbol{r}_k$ 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

$$S_C(X_1, X_2) = \frac{X_1 \cdot X_2}{\|X_1\|\|X_2\|} \tag{8}$$

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 calculate a cloud of 2D points

$\{S^i_C(B), S^i_C(d)\}$ with $S^i_C(B) = S_C\{B(j,\boldsymbol{r}_t), B(j,\boldsymbol{r}_s)\}$, $S^i_C(d) = S_C\{d(\epsilon,\boldsymbol{r}_t), d(\epsilon,\boldsymbol{r}_s)\}$ for $N_{\rm sim}$ points $\{t,s\}$ sampled from the simulation cell. We consider $N_{\rm sim} = 200\times200 = 40,000$ 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 $R_{\rm cut}$ and $J_{\rm max}$. We need to consider two limiting cases: (i) when $S^i_C(B) \ll S^i_C(d)$, even drastically dissimilar descriptors might yield the same LDOS, and modeling becomes trivial from a ML perspective. This case applies when $R_{\rm cut}$ 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 $R_{\rm cut}$ to be large, the descriptors will be physically well informed. But we risk approaching (ii) $S^i_C(B) \gg S^i_C(d)$, which makes our problem difficult to model in terms of ML. For a fixed choice of $R_{\rm cut}$, the number of expansion coefficients governed by $J_{\rm max}$ 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 $R_{\rm cut}$ and minimum $J_{\rm max}$ for which $S^i_C(B) \approx S^i_C(d)$ is the optimal choice.

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

$$\mathrm{ACSD} = \frac{1}{N_{\rm sim}} \sum_i^{N_{\rm sim}} \left| S^i_C(d) - S^i_C(B) \right| , \tag{9}$$

between all points within the 2D point cloud $\{S^i_C(B), S^i_C(d)\}$ and $\{S^i_C(B), S^i_C(B)\}$, where $N_{\rm sim}$ denotes the number of sample points. The ACSD thus measures the average over the distribution of similarities that deviate from the $S^i_C(B) = S^i_C(d)$ line. Eq. (9) 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 g/cc) [57].

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 $R_{\rm cut} = 4.676$Å and $J_{\rm max} = 5$ which were identified as accurate in Ref. [32]. 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 Fig. 6 for a fixed number of components $J_{\rm max}$ and varying cutoff radius $R_{\rm cut}$, and vice versa in Fig. 7.

It is evident in Fig. 6 that the NN prediction (green) becomes more accurate with increasing $R_{\rm cut}$, up until a minimum at 4.676 Å, after which a slight decrease is observed. This behavior is reproduced, almost exactly, by the ACSD surrogate metric (red). These results follow the intuitive expectation: a small $R_{\rm cut}$ 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 $R_{\rm cut}$ 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 differs. Consequently, we expect the optimal $R_{\rm cut}$ in between these extremes, which is correctly identified by the ACSD. Furthermore, Fig. 6 confirms the prior assertion that ML modeling becomes increasingly more challenging as $R_{\rm cut}$ increases, as is evident from the increasing spread in the network prediction errors.

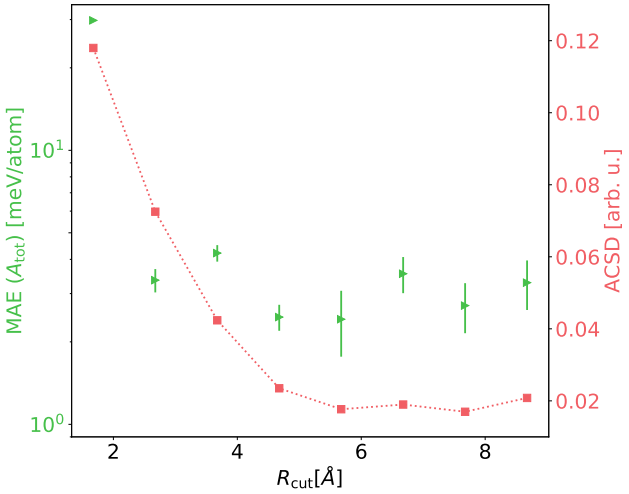

FIG. 6. NN errors when using SNAP descriptors with differing $R_{\rm cut}$. $J_{\rm max}$ was kept at 5.

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 $R_{\rm cut}$ and and varying $J_{\rm max}$ yields a similar trend which is illustrated in Fig. 7. As long as $J_{\rm max}$ is chosen sufficiently large, there is little effect on the accuracy of the NN (green). 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

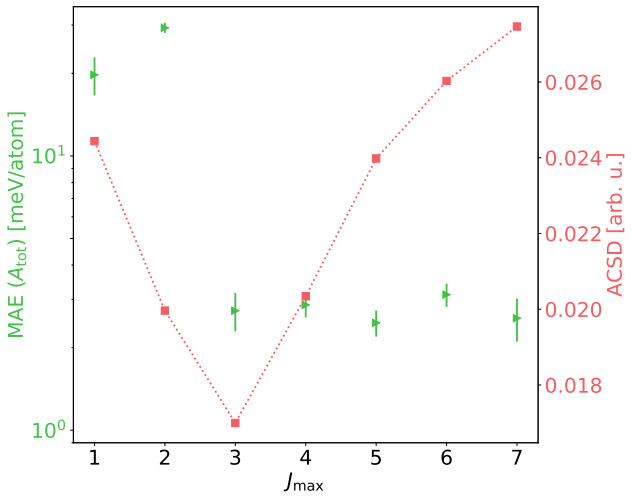

FIG. 7. NN errors when using SNAP descriptors with differing J_{max}. R_{cut} was kept at 4.676Å .

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 could choose the first data point for which the ACSD encounters a minimum, in this case $J_{\text{max}} = 3$. 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.

III. DISCUSSION

We tackle the complex optimization problem inherent to any ML 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 Fig. 1), 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 [36] 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.

By incorporating these two developments, we have devised a highly efficient ML surrogate modelling workflow shown in Fig. 1. 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.

METHODS

Density Functional Theory The electronic structure of the materials under investigation was calculated using Finite-Temperature Density Functional Theory (DFT) in the Kohn-Sham picture [4, 5]. All DFT calculations were carried out using the QuantumESPRESSO software library [58–60]. Suitable atomic configurations were obtained by coupling DFT simulations to a molecular dynamics (MD) simulation, i.e., a classical mechanical treatment of the ions. DFT-MD calculations were performed using either QuantumESPRESSO (beryllium systems) or VASP [61–63] (aluminum systems). For the beryllium data set, a $11 \times 11 \times 11$ k-grid has been used and for the aluminum data set a $8 \times 8 \times 8$ k-grid has been used. In either case, Monkhorst-Pack [64] sampling was employed. All DFT-MD calculations were performed at the Γ-point. All DFT calculations were carried out using a plane-wave basis set, with beryllium calculations using a cutoff energy of 40 Ry (both DFT and DFT-MD) and an ultrasoft pseudopotential [65, 66], while all aluminum DFT calculations were performed using a cutoff energy of 100 Ry and a norm-conserving Vanderbilt [67] pseudopotential. Aluminum DFT-MD calculations were performed using a 240 eV cutoff energy and a PAW pseudopotential [68, 69]. The PBE exchange-correlation functional [70] has been used for all beryllium calculations as well as for the aluminum DFT-MD calculation, while the PBEsol functional was used for the aluminum DFT calculations [71].

Neural Networks Neural networks are powerful regression models consisting of layers of so called neurons or perceptrons [72]. 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

$$\boldsymbol{y} = \sigma(\mathbf{W}\boldsymbol{x} + \boldsymbol{b}) \,, \tag{10}$$

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 neural network is tuned on an appropriate amount of data in a process called training [73, 74]. Such a training process is usually performed using gradient based methods in an iterative fashion, based on back-propagation [75]. 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*. Neural network performance is highly dependent on the correct choice of hyperparameters that characterize both architecture and training policy for a neural network, such as the number and width of individual layers. We use feed-forward neural networks, in which each neuron of a layer is connected to all neurons of subsequent layers, to map one vector quantity to another. Neural networks were constructed using the PyTorch library [76] and trained on a single GPU.

Hyperparameter optimization Hyperparameter optimization tackles Eq. 1. This can be done by a direct search algorithm, automated libraries, or custom algorithms. Hyperparameter optimization algorithms suggest *trial* or *candidate* NNs, which are trained and then judged and/or analyzed to determine an optimal set of hyperparameters. For this study, we employed the Optuna library, and provide our own implementations of the NASWOT and OAT algorithms in the MALA package. Details for these implementations and performed experiments are provided in the Supplementary Methods section and Supplementary Note 1.

DFT Surrogate Models Scalable DFT surrogate models are constructed using a workflow based on the local density of states $d(\epsilon, \boldsymbol{r})$ (LDOS) to compute quantities of interest. The LDOS of a system is a spatially and energy resolved intermediate quantity of electronic structure calculations. Surrogate models based on the LDOS are capable of accurately reproducing other intermediate quantities (electronic density $n(\boldsymbol{r})$, electronic density of state $D(\epsilon)$) as well as observables (total free energy, atomic forces). Most importantly, the fundamental calculation of the total free energy in Kohn-Sham DFT

$$A^{\mathrm{BO}}_{\mathrm{total}}[n] = T_{\mathrm{S}}[n] - k_{\mathrm{B}}\tau S_{\mathrm{S}}[n] + U[n] + E^{\tau}_{\mathrm{XC}}[n] + E^{ei}[n] \,, \quad (11)$$

can be replaced by

$$A^{\mathrm{BO}}_{\mathrm{total}}[d] = E_b[D[d]] - k_{\mathrm{B}}\tau S_{\mathrm{S}}[D[d]] - U[n[d]] + E^{\tau}_{\mathrm{XC}}[n[d]] - \int d\boldsymbol{r}\, v^{\tau}_{\mathrm{XC}}(\boldsymbol{r}) n[d](\boldsymbol{r}) \,, \quad (12)$$

which solely depends on the LDOS through the electronic density and density of states. Here, T_{S} denotes the Kohn-Sham kinetic energy of non-interacting fermions, S_{S} the Kohn-Sham entropy of non-interacting electrons, U the Hartree energy, E^{τ}_{XC} the exchange-correlation energy, E^{ei} the energy associated with the electron-ion interaction, $v^{\tau}_{\mathrm{XC}}(\boldsymbol{r})$ the exchange-correlation potential, τ the electronic temperature, and k_{B} the Boltzmann constant. During hyperparameter optimization, the band energy

$$E_b[d] = \int d\epsilon\, f^{\tau}(\epsilon)\epsilon D[d](\epsilon) \quad (13)$$

is used instead of $A^{\mathrm{BO}}_{\mathrm{total}}[d]$ for monitoring the intermediate accuracy of the NN. The mathematical foundation of this workflow is briefly outlined in the Supplementary Methods Section and explained in detail in Ref. [32].

Data and code availability All machine-learning experiments and post-processing analysis have been carried out with the MALA code, version 1.0.0. Training data and benchmark models for aluminum experiments can be found in [49] and for beryllium experiments in [57]. Code used to conduct hyperparameter optimization as well as the relevant models can be found in [77].

REFERENCES

[1] S. Kurth and J. P. Perdew, Int J Quantum Chem **77**, 814 (2000).
[2] R. M. Martin, *Electronic Structure: Basic Theory and Practical Methods* (Cambridge University Press, Cambridge, 2004).
[3] P. Hohenberg and W. Kohn, Phys. Rev. **136**, B864 (1964).
[4] W. Kohn and L. J. Sham, Phys. Rev. **140**, A1133 (1965).
[5] N. D. Mermin, Phys. Rev. **137**, A1441 (1965).
[6] M. Born and R. Oppenheimer, Ann Phys **389**, 457 (1927).
[7] J. Toulouse, arXiv:2103.02645 (2021).
[8] J. Dziedzic, A. Bhandari, L. Anton, C. Peng, J. C. Womack, *et al.*, J. Phys. Chem. C **124**, 7860 (2020).
[9] V. V. Karasiev, J. Hinz, S. X. Hu, and S. B. Trickey, Nature **600**, E12 (2021).
[10] A. Nakata, J. S. Baker, S. Y. Mujahed, J. T. L. Poulton, S. Arapan, J. Lin, *et al.*, J. Chem. Phys. **152**, 164112 (2020).
[11] M. W. C. Dharma-wardana, D. D. Klug, and R. C. Remsing, Phys. Rev. Lett. **125**, 075702 (2020).
[12] G. Massacrier, M. Böhme, J. Vorberger, F. Soubiran, and B. Militzer, Phys. Rev. Research **3**, 023026 (2021).
[13] T. J. Callow, E. Kraisler, S. B. Hansen, and A. Cangi, arXiv:2103.09928 (2021).
[14] L. Fiedler, K. Shah, M. Bussmann, and A. Cangi, arXiv:2110.00997 (2021).
[15] J. Wei, X. Chu, X.-Y. Sun, K. Xu, H.-X. Deng, J. Chen, *et al.*, InfoMat **1**, 338 (2019).
[16] J. E. Gubernatis and T. Lookman, Phys. Rev. Materials **2**, 120301 (2018).
[17] G. Carleo, I. Cirac, K. Cranmer, L. Daudet, M. Schuld, N. Tishby, *et al.*, Rev. Mod. Phys. **91**, 045002 (2019).
[18] J. Schmidt, M. R. G. Marques, S. Botti, and M. A. L. Marques, npj Comput Mater **5**, 83 (2019).
[19] Z. W. Ulissi, A. J. Medford, T. Bligaard, and J. K. Nørskov, Nat Commun **8**, 14621 (2017).
[20] K. T. Schütt, H. Glawe, F. Brockherde, A. Sanna, K. R. Müller, and E. K. U. Gross, Phys. Rev. B **89**, 205118 (2014).
[21] R. Ramakrishnan, P. O. Dral, M. Rupp, and O. A. von Lilienfeld, J. Chem. Theory Comput. **11**, 2087 (2015).
[22] V. L. Deringer and G. Csányi, Phys. Rev. B **95**, 094203 (2017).
[23] G. C. Sosso, G. Miceli, S. Caravati, J. Behler, and M. Bernasconi, Phys. Rev. B **85**, 174103 (2012).
[24] T. Morawietz and J. Rg Behler, J. Phys. Chem. A (2013), 10.1021/jp401225b.
[25] A. P. Bartók, M. C. Payne, R. Kondor, and G. Csányi, Phys. Rev. Lett. **104**, 136403 (2010).

[26] J. Schmidt, J. Shi, P. Borlido, L. Chen, S. Botti, and M. A. L. Marques, Chem. Mater. **29**, 5090 (2017).
[27] J. S. Smith, O. Isayev, and A. E. Roitberg, Chem. **8**, 3192 (2017).
[28] J. C. Snyder, M. Rupp, K. Hansen, K.-R. Müller, and K. Burke, Phys. Rev. Lett. **108**, 1079 (2012).
[29] F. Brockherde, L. Vogt, L. Li, M. E. Tuckerman, K. Burke, and K.-R. Müller, Nat Commun **8**, 872 (2017).
[30] M. Tsubaki and T. Mizoguchi, Phys. Rev. Lett. **125**, 206401 (2020).
[31] A. Chandrasekaran, D. Kamal, R. Batra, C. Kim, *et al.*, npj Comput Mater **5**, 22 (2019).
[32] J. A. Ellis, L. Fiedler, G. A. Popoola, N. A. Modine, J. A. Stephens, A. P. Thompson, *et al.*, Phys. Rev. B **104**, 035120 (2021).
[33] S. P. Huber, S. Zoupanos, M. Uhrin, L. Talirz, L. Kahle, R. Häuselmann, *et al.*, Sci. Data **7**, 300 (2020).
[34] A. H. Larsen, J. J. Mortensen, J. Blomqvist, I. E. Castelli, R. Christensen, M. Du\lak, *et al.*, J. Phys. Condens. Matter **29**, 273002 (2017).
[35] M. Uhrin, S. P. Huber, J. Yu, N. Marzari, and G. Pizzi, Comput. Mater. Sci. **187**, 110086 (2021).
[36] J. Mellor, J. Turner, A. Storkey, and E. J. Crowley, arXiv:2006.04647v1 (2020).
[37] K. Chauhan, S. Jani, D. Thakkar, R. Dave, J. Bhatia, S. Tanwar, *et al.*, in *2020 IEEE Int. Conf. Innov. Mech. Ind. Appl. ICIMIA 2020 - Proc.* (2020) pp. 205–212.
[38] A. Cangi, J. A. Ellis, L. Fiedler, D. Kotik, N. A. Modine, *et al.*, "MALA," (2021).
[39] F. Hutter, J. Lücke, and L. Schmidt-Thieme, Künstl Intell **29**, 329 (2015).
[40] A. P. Thompson, L. P. Swiler, C. R. Trott, S. M. Foiles, and G. J. Tucker, J. Comput. Phys. **285**, 316 (2015).
[41] M. A. Wood and A. P. Thompson, J. Chem. Phys **148**, 241721 (2018).
[42] M. A. Wood, M. A. Cusentino, B. D. Wirth, and A. P. Thompson, Phys. Rev. B **99**, 184305 (2019).
[43] M. A. Cusentino, M. A. Wood, and A. P. Thompson, J. Phys. Chem. A **124**, 5456 (2020).
[44] R. Hooke and T. A. Jeeves, J. ACM **8**, 212 (1961).
[45] R. M. Lewis, V. Torczon, and M. W. Trosset, J. Comput. Appl. Math. **124**, 191 (2000).
[46] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, Adv. Neural Inf. Process. Syst. **24** (2011).
[47] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, in *Proc. ACM SIGKDD Int* (Association for Computing Machinery, 2019) pp. 2623–2631.
[48] X. Zhang, X. Chen, L. Yao, C. Ge, and M. Dong, in *Neural Information Processing*, edited by T. Gedeon, K. W. Wong, and M. Lee (Springer International Publishing, Cham, 2019) pp. 287–295.
[49] J. A. Ellis, L. Fiedler, G. A. Popoola, N. A. Modine, J. A. Stephens, *et al.*, "LDOS/SNAP data for MALA: Aluminium at 298K and 933K," (2021).
[50] M. G. KENDALL, Biometrika **30**, 81 (1938).
[51] A. P. Bartók, R. Kondor, and G. Csányi, Phys. Rev. B **87**, 184115 (2013).
[52] M. Rupp, A. Tkatchenko, K.-R. Müller, and O. A. Von Lilienfeld, Phys. Rev. Lett. **108**, 058301 (2012).
[53] K. Hansen, F. Biegler, R. Ramakrishnan, W. Pronobis, O. A. v. Lilienfeld, K.-R. Müller, *et al.*, J. Phys. Chem. Lett. **6**, 1948 (2015).
[54] F. A. Faber, A. S. Christensen, B. Huang, and O. A. von Lilienfeld, arXiv:1712.08417 (2017).
[55] R. Drautz, Phys. Rev. B **99**, 014104 (2019).
[56] Y. Lysogorskiy, C. van der Oord, A. Bochkarev, S. Menon, M. Rinaldi, T. Hammerschmidt, *et al.*, npj Comput Mater **7**, 1 (2021).
[57] L. Fiedler and A. Cangi, "LDOS/SNAP data for MALA: Beryllium at 298K," (2022).
[58] P. Giannozzi, O. Baseggio, P. Bonfà, D. Brunato, R. Car, I. Carnimeo, *et al.*, J. Chem. Phys. **152**, 154105 (2020).
[59] P. Giannozzi, S. Baroni, N. Bonini, M. Calandra, R. Car, *et al.*, J. Phys.: Condens. Matter **21**, 395502 (2009).
[60] P. Giannozzi, O. Andreussi, T. Brumme, O. Bunau, M. B. Nardelli, *et al.*, J. Phys. Condens. Matter **29**, 465901 (2017).
[61] G. Kresse and J. Hafner, Phys. Rev. B **47**, 558 (1993).
[62] G. Kresse and J. Furthmüller, Phys. Rev. B **54**, 11169 (1996).
[63] G. Kresse and J. Furthmüller, Comput. Mater. Sci. **6**, 15 (1996).
[64] H. J. Monkhorst and J. D. Pack, Phys. Rev. B **13**, 5188 (1976).
[65] A. D. Corso, Comput. Mater. Sci. **95**, 337 (2014).
[66] K. Lejaeghere, G. Bihlmayer, T. Björkman, P. Blaha, S. Blügel, V. Blum, *et al.*, Science **351**, aad3000 (2016).
[67] D. R. Hamann, Phys. Rev. B **88**, 085117 (2013).
[68] P. E. Blöchl, Phys. Rev. B **50**, 17953 (1994).
[69] G. Kresse and D. Joubert, Phys. Rev. B **59**, 1758 (1999).
[70] J. P. Perdew, K. Burke, and M. Ernzerhof, Phys. Rev. Lett. **77**, 3865 (1996).
[71] J. P. Perdew, A. Ruzsinszky, G. I. Csonka, O. A. Vydrov, G. E. Scuseria, L. A. Constantin, *et al.*, Phys. Rev. Lett. **100**, 136406 (2008).
[72] F. Rosenblatt, *The Perceptron: A Perceiving and Recognizing Automaton (Project PARA).*, Tech. Rep. 85-460-1 (Cornell Aeronautical Laboratory, 1957).
[73] K. Hornik, M. Stinchcombe, and H. White, Neural Netw **2**, 359 (1989).
[74] K. Hornik, Neural Netw **4**, 251 (1991).
[75] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning Book*, illustrated edition ed. (The MIT Press, Cambridge, Massachusetts, 2016).
[76] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, *et al.*, in *Adv. Neural Inf. Process. Syst.*, Vol. 32 (Curran Associates, Inc., 2019).
[77] L. Fiedler, N. Hoffmann, P. Mohammed, T. Yovell, V. Oles, J. A. Ellis, *et al.*, "Scripts and networks for "Electronic Structure Machine Learning Surrogates without Training"," (2022).

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 Ad-

ministration 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.

AUTHOR CONTRIBUTIONS

L.F. performed the beryllium DFT(-MD) calculations, all ML experiments except for the direct search and implemented the Optuna interface and surrogate metrics for the SNAP descriptors. The NASWOT code was implemented by N.H. and L.F., while P.M and L.F. implemented the OAT code. T.Y. calculated data necessary for the SNAP surrogate metric analysis. G.A.P performed the direct search for the aluminum model. V.O. investigated the scalability of the Optuna interface. J.A.E implemented the direct search algorithm used for the aluminum model and the development of the surrogate metric investigated for the SNAP descriptors. S.R. contributed to theory and initial development of the MALA framework. A.C. contributed to the theory and development of the MALA framework, supported data analysis, and supervised the overall project. All authors contributed to writing the manuscript.