Faraday Discussions

Cite this: *Faraday Discuss.*, 2025, **256**, 483

ROYAL SOCIETY OF CHEMISTRY

PAPER

View Article Online
View Journal | View Issue



# How big is big data?

Daniel Speckhard, *[ab] Tim Bechtel, [ab] Luca M. Ghiringhelli, [c] Martin Kuban, [a] Santiago Rigamonti [a] and Claudia Draxl [ab]

Received 14th May 2024, Accepted 8th July 2024
DOI: 10.1039/d4fd00102h

Big data has ushered in a new wave of predictive power using machine-learning models. In this work, we assess what big means in the context of typical materials-science machine-learning problems. This concerns not only data volume, but also data quality and veracity as much as infrastructure issues. With selected examples, we ask (i) how models generalize to similar datasets, (ii) how high-quality datasets can be gathered from heterogenous sources, (iii) how the feature set and complexity of a model can affect expressivity, and (iv) what infrastructure requirements are needed to create larger datasets and train models on them. In sum, we find that big data present unique challenges along very different aspects that should serve to motivate further work.

## 1 Introduction

Big data is a term that not only governs social media and online stores, but has entered most modern research fields. As such, it also concerns materials science. While our research has always been largely based on data, through the analysis and interpretation of measured and computed results, in recent times, more aspects have come into play. For instance, there is the practical reason that more and more funding bodies require data to be kept for a certain period of time. On the scientific side, the idea of sharing data is enjoying popularity, thus avoiding the same investigation being done multiple times. Most important is the use of data in machine learning (ML), or more generally, artificial intelligence (AI).

Artificial intelligence is arguably the most rapidly emerging topic in various research domains, with increasing impact in materials science as well. The success of AI approaches, however, strongly depends on the amount and quality of the underlying training data. In this context, the first obvious question is how large a dataset needs to be in order to provide sufficient information for the research problem to be solved. Are millions of scientific results, as available in

[a]*Physics Department and CSMB, Humboldt-Universität zu Berlin, Zum Großen Windkanal 2, 12489 Berlin, Germany. E-mail: claudia.draxl@physik.hu-berlin.de; dts@physik.huberlin.de; Fax: +49 2093 66361; Tel: +49 2093 66363*

[b]*Max Planck Institute for Solid State Research, Heisenbergstraaae 1, 70569 Stuttgart, Germany*

[c]*Department of Materials Science and Engineering, Friedrich-Alexander Universität Erlangen-Nürnberg, Dr.-Mack-Str. 77, 90762 Fürth, Germany*

international databases, a gold mine? Or do these collections still not provide enough significant data for a specific question? Or can we even learn from small datasets? Are large, multipurpose databases sufficient for training models, or to what extent are dedicated datasets needed for this task? Can errors be controlled when using data from different sources? Can we learn from experimental and theoretical data together, or are even different theoretical or experimental data by themselves too heterogeneous to produce reasonable predictions?

What is useful for a particular learning task may depend heavily on the research question, the quantity to be learned, and the methodology employed. This concerns data quality, data interoperability – especially when data from different sources are brought together – data veracity, and data volume (the last two are part of the "Four V's of big data"[1]). In this work, we ask the question: "What does big mean in the realm of materials science data?" There are many issues related to this, demonstrated by a few examples: first, some methods are more data hungry than others. While, for instance, symbolic regressors may lead to reasonable results already for a small data set,[2] neural networks (NNs) may require many more data points[3] for training to be stable. Second, even with what can be considered big datasets, training models that generalize to similar datasets is not trivial. Third, data quality may have a significant impact. While high-quality data may give one a clear picture from the very beginning, many more noisy data may need to be accumulated for obtaining robust results, to avoid wrong conclusions and/or allow for physical interpretation. Finally, what does big data mean for data infrastructure? What are the requirements for processing and storing big datasets; how computationally intensive is training ML models on them?

In this work, we address such questions and evaluate the performance of different AI models and tools in terms of data volume, diversity and quality, and provide a quantitative analysis. As the overall topic is very wide and diverse, we aim to draw the reader's attention to it and initiate discussions rather than to provide detailed solutions to all of the related issues. The chosen examples should serve this purpose. In the first one, we ask whether a model obtained from a message-passing neural network (MPNN) can be transferred to a similar dataset. Second, similarity metrics and sorting are applied to understand data quality in terms of the convergence behavior of density-functional theory (DFT) data. Third, the effect of an expanded feature set on the expressivity of cluster expansion is demonstrated. Fourth, the complexity of different model classes is explored in terms of their performance. Finally, infrastructure requirements for creating large materials datasets and for training models with many parameters are quantitatively analyzed.

## 2 Transferability of models

The Materials Project (MP)[4] and AFLOW[5] are popular data collections from computational materials science. Both are also often used for ML purposes. Most of the calculations are carried out *via* DFT with the same code (VASP),[6] projector augmented wave method) and exchange–correlation functionals (mostly the generalized gradient approximation in the PBE[7] parametrization). Since both databases also contain a significant number of materials that are common to both, *i.e.*, that share the same composition and spacegroup, we ask the question

whether a model trained on one dataset would perform well when making inferences on the other dataset. We choose the formation energy as our target, since it is a rather well-behaved property, only relying on the total energies of the compounds and their constituents.

The AFLOW dataset is chosen as described in ref. 8 and filtered to use only calculations performed with the PBE functional, which contain both bandgap and formation energy. The MP dataset is retrieved from ref. 9 (dataset snapshot denoted as MP-crystals-2018.6.1). After filtering, the two datasets comprise 61 898 and 60 289 materials, respectively. We proceed by assigning unique identifiers to the structures in the AFLOW and MP datasets. They consist of the chemical formula concatenated with the space group of the crystal, similar to ref. 10, to define which structures are common to both databases and which are unique to either of them. This results in 50 652 unique composition–spacegroup pairs in the AFLOW dataset and 54 438 in the MP dataset, with 8591 pairs being common to both datasets.

To avoid leakage of information from the training set to the test set, we ensure that the same common structures are present in the training-test splits of both datasets. This means that a model trained on AFLOW data will not be used for inference on MP test data on the same structure it was trained on, and *vice versa*. For instance, all structures with composition $Mg_2F_4$ and spacegroup 136 get the label Mg2F4_136. If there are several of these structures, *e.g.*, with different lattice parameters, all of them get assigned to one split, *i.e.*, training, validation, or test.

We train MPNNs with edge updates, as described in ref. 11, on the two individual datasets as well as the combination thereof. Previously, we have shown that the hyperparameters that define the architecture of the MPNN transfer well to different prediction tasks.[8] Following this, we perform no further hyperparameter optimization. The performance of the models is shown in Fig. 1. The plots along the diagonal show model performance when training and test data are from the same database. Let us focus first on AFLOW and MP data ($2 \times 2$ submatrix on the bottom left). Both models trained and evaluated on the individual databases are very good, with mean absolute errors (MAEs) of 30 meV per atom and 23 meV per atom, respectively. However, the errors in the prediction of the other database are an order of magnitude larger. Notably, there is also a clear trend of too large (small) predictions in the MP-model predictions on AFLOW (AFLOW-model predictions on MP) data. This points to a systematic offset between the energies of the two datasets. Indeed, we can partly trace this discrepancy back to the relatively small overlap of the spacegroup–composition pairs in the two databases, which is about 16% only. Further analysis reveals that MP data contain many more materials with lower formation energies than the AFLOW data, as evident from the top panel of Fig. 2. The distribution of the shared structures (bottom panel of Fig. 2) indicates that the latter have overall a smaller impact, *i.e.*, do not lead to a systematic offset. However, we see that there are many more entries of the same structures, *e.g.*, with different lattice parameters, in the AFLOW dataset. For example, $Ti_2O_4$ (spacegroup 136) and $BaTiO_3$ (spacegroup 221) occur 127 and 115 times, respectively, while only one of each compound is found in the MP data.

The lack of interoperability of the two datasets is also demonstrated when training a model on the combined dataset (top right panel in Fig. 1). Although there is an improvement over the models transferred to the respective other

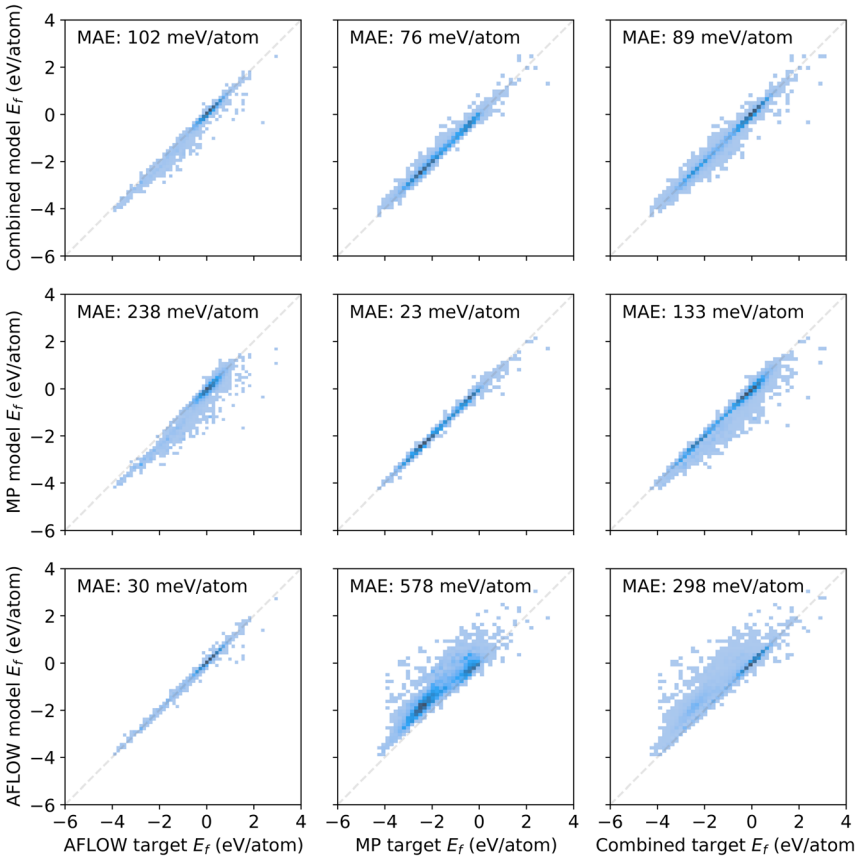


Fig. 1 Predicted *versus* calculated (target) formation energies for AFLOW and MP data, as well as the combined dataset. The bottom row (left column) shows the model trained (evaluated) on the AFLOW data, the middle row (middle column) the MP data, and the top row (right column) the combined data.

individual dataset, it performs worse than the models that are trained and tested on either of them (bottom left, center middle). We assign the discrepancy to differences in computational details, such as Brillouin-zone (BZ) sampling, basis-set cutoff, convergence criteria, *etc.*

Even for what we might consider to be big datasets in materials science, and using a rather simple ML target, this example shows us that the models we train are only valid for the data they were trained on and struggle to generalize. In our specific example, the AFLOW and MP datasets turn out to sample the underlying material space differently, since the MP materials appear biased towards lower formation energies. We conclude that these two databases are not "big" enough in the sense that they are not diverse enough to be able to make predictions across a wider range of diversity. Training on the combined dataset, the predictions are less biased, *i.e.*, the errors are more symmetric with respect to the diagonal. However, the MAE is higher overall as compared to the individual models. This indicates that also differences in computational settings may play a role and may need to be considered as input features.

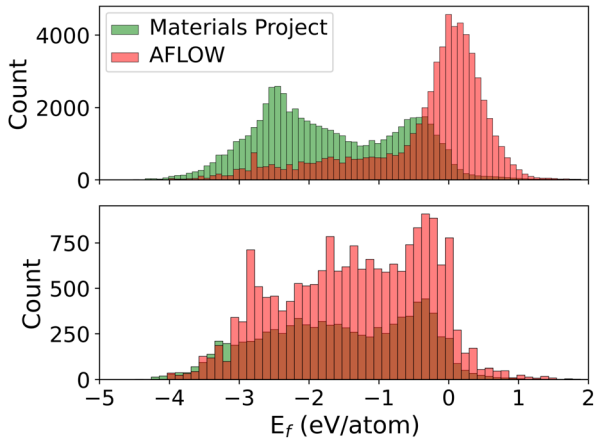


Fig. 2 Top: distribution of formation energies in the two datasets containing 50 652 (AFLOW) and 54 438 (MP) unique composition–spacegroup pairs. Bottom: distribution of formation energies for the 8591 composition–spacegroup pairs that are present in both datasets. Data from the AFLOW dataset with $E_f > 2$ eV per atom are not visible on this scale and are thus cut off in this representation.

## 3 Revealing data quality *via* similarity measures

Veracity, another of the "Four V's of big data", poses a challenge to ML applications by introducing a fundamental level of noise that cannot be overcome with more complex models. An example from computational materials science is sets of calculations with different levels of accuracy[12] determined by the chosen approximation, such as the exchange–correlation (xc) functional of DFT, or different levels of numerical precision, determined by input parameters, such as the basis-set size or *k*-grid. Related to this is a practical problem: if multiple calculations from different sources – with smaller or larger deviations in the results for a particular physical property – are available, which value should be trusted? This challenge can be met by either applying corrections to the data to bring them onto the same level of accuracy/precision[13] or by filtering them for consistent subsets. The former option requires additional ML models,[2] trained on dedicated, high-precision datasets. However, generating the required data for training these models is costly, and making predictions for materials with large unit cells may require that the models are trained on such systems as well. Filtering the data by their numerical precision, on the other hand, can be applied to existing datasets, but one may not find enough data for a particular application because the number of calculations with exactly the same numerical settings is typically small. The number of calculations that can be used together can be increased, however, if we can understand – and quantify – the level of convergence of data with the computational parameters. Understanding and visualizing the convergence behavior of computed properties can be achieved by introducing similarity metrics, as recently demonstrated with the example of fingerprints of the electronic density of states (DOS).[14,15] Given a large enough set of calculations for a single material, the relationships between the parameters used in calculations and the similarities of the results can be shown.

To illustrate this, we make use of data from the NOMAD data collection.[16,17] These calculations were carried out with the DFT code FHI-aims[18] as part of a systematic study of the impact of computational parameters on DFT results.[19] For our analysis, we use the calculations for hexagonal boron nitride (h-BN), specifically, the DOS of ground-state calculations obtained with different basis-set sizes and $k$-point samplings at the experimentally determined equilibrium volume. To compare the results of these calculations, we compute the spectral fingerprints[20] of the DOS in the energy range between −10 and 10 eV around the valence band maximum. In Fig. 3, it is exemplified how the similarity between two spectra is obtained. The left panels show the DOS obtained by two calculations with different numbers of basis functions ($N_b$), but otherwise identical settings. They are converted into spectral fingerprints (middle panels), *i.e.*, raster-image representations of the original spectra. The similarity (right panel) is calculated using the Tanimoto coefficient,[21] which is the overlap of the areas covered by the individual fingerprints (red and blue) divided by the total area covered by both of them (purple). The similarity matrix shown in Fig. 4 contains all pairwise similarities. The rows and columns are sorted by increasing numerical settings, separately for two xc functionals, the local density approximation (LDA, indices ≤70) and PBE (indices >70). The bottom panel shows the number of $k$-points ($N_{kpt}$) and $N_b$. For each $k$-point mesh (plateau in the $k$-mesh), the basis set is increased in the same way. Additionally, the data are sorted by a set of numerical settings, called "light", "tight", and "really tight" in FHI-aims. Finally, consecutive calculations with otherwise identical settings vary by the relativistic approximation employed for core electrons, *i.e.*, "ZORA", and "atomic ZORA".[18] Sorting the matrix in this way, we see patterns appearing in the figure, which we discuss below. As a guide to the eye, dotted lines inside the matrix indicate sets of calculations with the same number of $k$-points. To simplify the discussion of the results, individual blocks are labeled with letters.

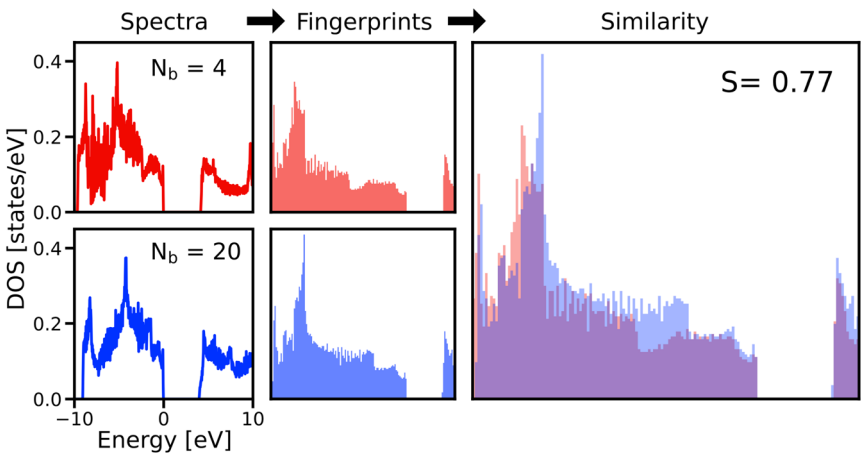


Fig. 3 Workflow for the calculation of DOS similarity. The two spectra (left panels), computed with different basis-set size $N_b$, are converted to fingerprints (middle panels), for which the similarity $S$ is calculated (right panel).

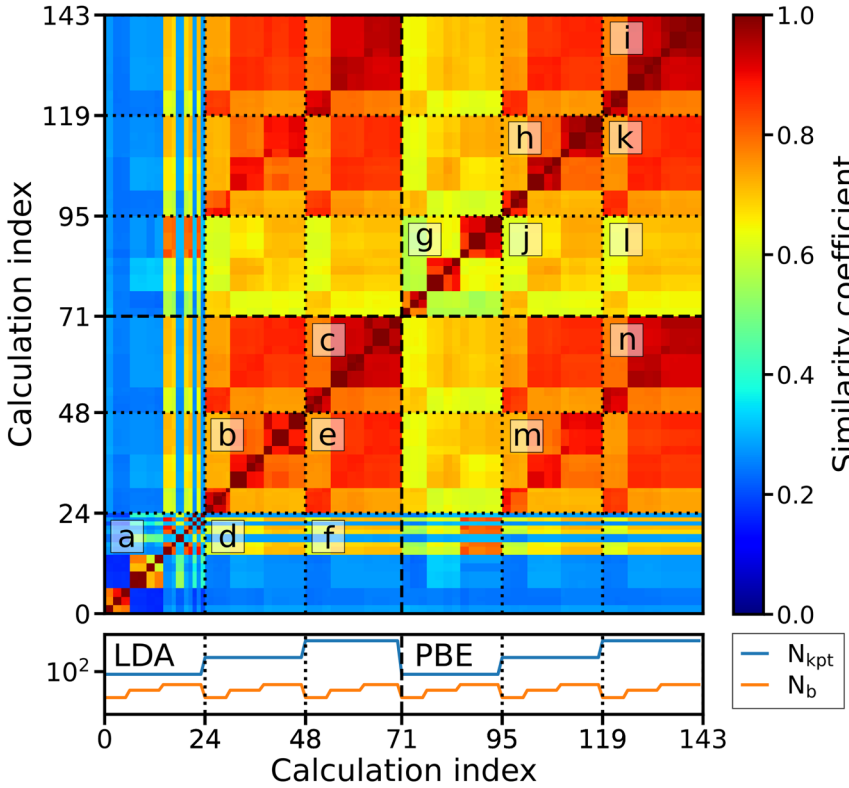


Fig. 4 DOS similarity matrices for h-BN obtained with different basis-set sizes and $k$-meshes, and two different xc functionals. The data are sorted such that low indices (≤70) correspond to LDA and high indices (>70) to PBE data. The color code indicates the similarity coefficient. Dotted lines group sets of calculations with the same $k$-point mesh. Letters mark blocks of the matrix that are discussed in the text. The bottom panel shows the number of $k$-points (blue) and the number of basis functions (orange).

Focusing first on the convergence of the LDA data (indices $i \leq 70$), we observe a clear block structure, where the calculations of the first set, *i.e.*, those with the lowest $N_{kpt}$ (index $i \leq 23$) are most dissimilar to all others (blocks d and f in Fig. 4), and also to each other (block a). However, they are pairwise similar, indicating that the relativistic approximation plays a minor role in the convergence of the DOS. Sets of calculations with medium ($24 \leq i \leq 47$) and high ($48 \leq i \leq 70$) numbers of $k$-points generally show higher similarity among themselves (blocks b and c). Noticeably, calculations with a low (medium and high) number of basis functions $N_b$ are similar among different BZ samplings (block e), but less similar to calculations with higher (lower) $N_b$. For PBE calculations $i \geq 71$, the convergence behavior is somewhat different: Already calculations with low $N_{kpt}$ reach medium similarity to calculations with high $N_{kpt}$ (j and l). Calculations with the same $N_b$ are similar to each other, already for low $N_{kpt}$ (block g). Similar to the LDA case, PBE calculations with medium $N_{kpt}$ reach high similarity to those with highest $k$-point density (block k). In the remaining diagonal blocks of the PBE data, *i.e.*, blocks h and i, it can be seen that the similarity between calculations

with the same $N_b$ is high and varies only slightly. Further comparing the similarity of LDA and PBE convergence, *i.e.*, the off-diagonal blocks m and n of the matrix (indices $i \geq 71$ on the $x$-axis and $i \leq 72$ on the $y$-axis), we find a pattern corresponding to calculations with the same number of basis functions. It shows that calculations with different xc functionals are more similar to each other if the same $N_b$ is used.

Our example illustrates the veracity of materials data arising from the large combinatorial space of approximations and numerical parameters employed in DFT codes. While LDA and PBE, both being semi-local functionals, are generally expected to give similar results in terms of the electronic structure, we show here that the convergence behavior with the number of $k$-points and basis functions is slightly different. The visualization of DOS similarity matrices allows one to understand and quantify differences between DFT data computed with different computational settings. It also enables the creation of rules to group data in order to gather "homogeneous" subgroups within a dataset.

# 4 Striving for expressivity in cluster expansion

In this example, we investigate the role of the feature set in the modeling of the bandgap of a family of perovskite materials by means of the cluster-expansion (CE) technique. CE is a multi-scale method based on a lattice Hamiltonian,[22] which is used to model and predict properties of alloys at different compositions and temperatures. In CE, an alloy configuration is represented by a vector $\mathbf{s}$ of occupation variables $s_i$. For a binary alloy, for example, the occupation variables may take the values $s_i = -1$ or 1, indicating the presence of one or the other species at crystal site $i$. By defining the cluster functions $\Gamma_c(\mathbf{s}) = \Pi_{i \in c} s_i$, where $c$ is a subset of crystal sites or cluster, it can be demonstrated[22] that any configuration-dependent property, can be formally expanded in terms of the symmetry-averaged cluster functions $X_c(\mathbf{s}) = \langle \Gamma_c(\mathbf{s}) \rangle$, named cluster correlations, where $\langle \cdot \rangle$ represents the symmetry average over all clusters symmetrically equivalent to $c$. For modeling the bandgap, this reads

$$E_g(\mathbf{s}) = \sum_c \lambda_c X_c(\mathbf{s}) \tag{1}$$

where $\lambda_c$ are the expansion coefficients. Here, the sum runs over a set containing $p$ symmetrically inequivalent clusters. In what follows, we call this family of models "Standard CE". Cluster expansion can be viewed as a ML problem,[23,24] where the cluster correlations $X_c$ are input features describing the structure of the crystal. This view can be leveraged by considering basis expansions based on these features, leading to nonlinear CE models[24] defined as

$$E_g(s) = \sum_m \theta_m h_m(\mathbf{X}(s)) \tag{2}$$

with the new expansion coefficients $\theta_m$, the (possibly nonlinear) functions $h_m(\mathbf{X}) : \mathbb{R}^p \to \mathbb{R}$ and $\mathbf{X}$ a $p$-vector with components $X_c$. The sum runs over a set containing $q$ functions $h_m$. For this example, we choose nonlinear polynomial features: every function $h_m(\mathbf{X})$ is a monomial up to degree $d$, *e.g.*, $h_j(\mathbf{X}) = {X_1}^2 X_2$ ($d = 3$). The total number of features in this case is given by $\sum_{k=0}^{d} \binom{p+k-1}{k}$.[24]

In this section, we want to explore the expressiveness of the feature spaces in standard and nonlinear CE. Assuming that the intrinsic error in the data to be modeled is small, does our feature space allow us to approximate the ground truth? We use a dataset of 235 oxide perovskites with composition $BaSnO_{3-x}$, with $x < 1$ being the number of oxygen vacancies per formula unit. $BaSnO_3$ has significant potential for use in photocatalytic and optoelectronic applications. Its electronic structure demonstrates a complex dependence on the concentration and configuration of oxygen vacancies. The bandgap varies strongly, even among structures with identical concentration and comparable energies of formation. Thus, modeling the bandgap is challenging for linear regression models. For this learning task, the clusters account for specific structural patterns of O vacancies.

As shown in Fig. 5, standard CE models, obtained *via* orthogonal matching pursuit (OMP) from a pool of $p = 27$ clusters, yield fit root-mean-squared errors (RMSEs) larger than 250 meV (blue line, top-left panel). The best fit is obtained, as expected, when using all 27 features (marked by a vertical dashed line). For the pool of $p = 27$ clusters, we explore polynomials of degree $d = 2$ and $d = 3$, containing 405 and 4059 features, respectively, obtaining models with up to 200 features using OMP. These are shown by the orange and red lines in the top-left panel of Fig. 5. In addition to the expected monotonic reduction of the error, two

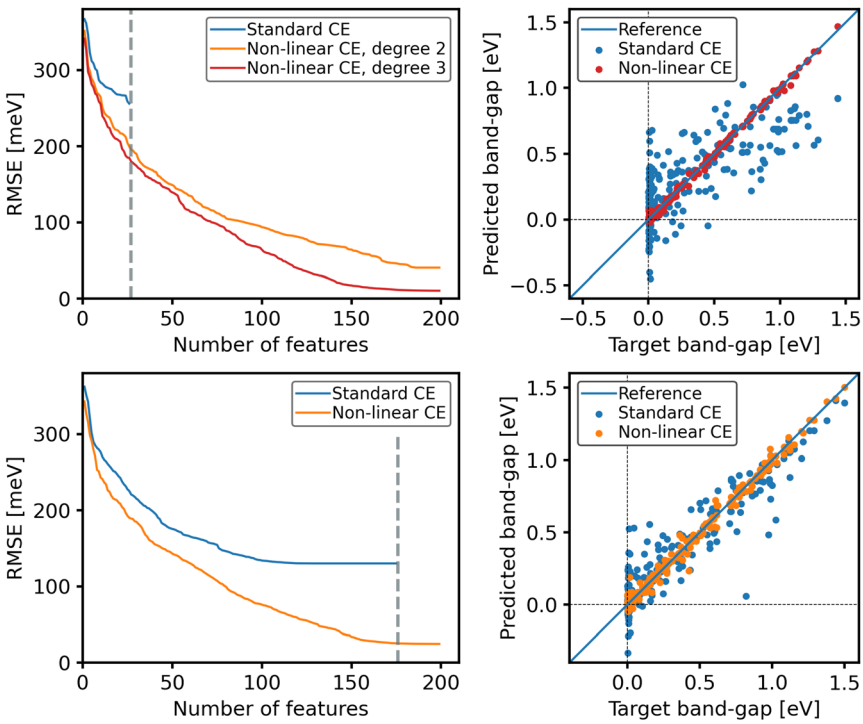


Fig. 5 Left: fitting errors for standard (blue) and nonlinear CE models (orange and red) based on a pool of 27 clusters (top) and 176 clusters (bottom). Right: predicted *versus* target bandgaps for the standard CE with 27 clusters and the nonlinear CE of degree 3 models with 150 features (top), and standard CE and nonlinear CE models with 150 clusters/features (bottom).

notable findings are obtained: first, with 27 features, the nonlinear feature space yields better models than the standard CE with the same number of 27 clusters. This is remarkable because the nonlinear features of the nonlinear CE models are derived solely from the 27 cluster correlations used in the standard CE model. Related to this point, we also observe that for the same number of features, degree 3 always gives smaller error than degree 2. Second, both nonlinear CE models reach a point where no further improvement is obtained upon increasing the number of features. This is observed as a plateau above ~150 features for $d = 3$ and above ~180 features for $d = 2$. The plateau for $d = 2$ is reached at a higher level of the RMSE than for $d = 3$. Thus, under the assumption that the intrinsic error is small, increasing the complexity of the feature space allows the CE to better approach the underlying ground truth. In practice, increasing the complexity of the feature space allows CE to obtain the same accuracy with sparser models as is obtainable with less complex feature spaces. This is important for model selection methods that favor sparsity, such as LASSO. The top-right panel of Fig. 5 shows the quality of the predictions for the whole dataset using the best possible model for standard CE (blue dots) and the best possible model for $d = 3$ nonlinear CE, based on the same 27 cluster correlations. The improvement of the nonlinear modeling is remarkable, especially for the difficult case of metals (the set of materials with $E_g = 0$).

Now, the question arises whether the beneficial effect of nonlinear features could be also obtained by adding more cluster correlations, that is, accounting for more structural patterns of vacancies. For this, we have created a pool of 176 clusters. The result of standard CE is shown in the lower left panel of Fig. 5 (blue line). Like before, we see a plateau setting in at about 110 clusters with a RMSE of ~125 meV, and no further improvement can be achieved upon adding more clusters. The vertical dashed line indicates the model with the largest possible number of clusters in this pool. Again, the second-order nonlinear CE based on this same pool allows for obtaining better fits at all numbers of features. A comparison of the quality of the fits for standard and nonlinear CE models with 150 clusters, respectively, is shown in the lower-right panel of the figure. Similarly to what was obtained for the small pool of 27 clusters, the nonlinear CE yields significantly better predictions.

From this example, we see the feature set can have a strong impact on the expressiveness of the CE model. Including nonlinear terms enables the CE to better fit the underlying ground truth even with a small data set size. This benefit appears irrespective of the number of cluster correlations. This improvement motivates us to try to better define model complexity, which is done in the following section.

# 5 Model complexity

Can knowing the complexity of the model tell us something about model performance for a given dataset? The term model complexity is not well defined in the literature. It describes the capacity of a model to learn an underlying probability distribution. We start this section by defining what complexity means for several model classes. We then evaluate how complexity determines model performance for two example datasets. Finally, we discuss the search for a model-independent scalar that represents model complexity.

### 5.1 Complexity for neural networks, random forest, and SISSO models

Here, we discuss how complexity can be quantified for three different types of models, *i.e.*, neural networks (NNs), the sure independence screening sparsifying operator (SSISO),[25] and random forests (RFs). For NNs, the total number of trainable parameters is often used as a measure for the model's complexity. That said, some of the NN parameters have a very different impact. The output of node $j$ at layer $k + 1$ in a feed-forward network is:

$$x_j^{k+1} = \sum_i \sigma\left(W_{ij} x_i^k + b_j\right) \tag{3}$$

This node receives an input $x$ from layer $k$, which is multiplied by a weight parameter $W_{ij}$ and summed with a bias parameter $b_j$ before being fed into a nonlinear activation function, $\sigma$. Thereby, the bias parameter is quite different from the weight parameter. So, a better description of complexity for NNs is a two-dimensional vector, containing the number of weights and biases. The benefit of this definition is that it is simple to compute. The drawback is that it is not a unique definition and does not differentiate between parameters deeper in the network, which are more expressive because they transform nonlinear inputs with nonlinear functions.[26]

The SISSO model first combines primary features into generated features using a set of mathematical operations (*e.g.*, +, −, exp(), √,..., and combinations thereof).[25,27] The number of features that are selected by the algorithm is called the dimension of the model. We can interpret this in terms of a (symbolic) tree that describes a generated feature, where each split in the tree is an operation and the leaves are the primary features.[27] The larger the depth of the tree, the more operations there are in the generated-feature space. The maximum tree depth is called the rung of the model. In essence, the number of selected generated features (model dimension) defines how many coefficients the model has. One can also include a single constant bias term in SISSO. With a larger rung value, the possible secondary features explode combinatorially. Therefore, we can define a descriptor for the complexity in SISSO with a two-dimensional vector, comprising the rung and the dimension of the model (counting the bias term if present as an extra dimension).

How can we define complexity for RFs? RFs are piecewise constant functions. For a regressor, each leaf node in each decision tree learns a constant bias term. For each split in each decision tree, a value of the variable to be split on must be learned. Since the trees are binary, the number of leaf nodes is one greater than the number of splits. Therefore, the number of leaf nodes is an integer that tells us the number of trainable parameters in the model and describes the complexity of the model.

From each of these three model types, we have presented a measure of model complexity. The model complexity for each model type is described with a different number/vector. For NNs, it is a two-dimensional vector of the total number of weights and biases; for SISSO, it is a two-dimensional vector reporting the rung and dimension of the model, while for RFs, it is the number of leaves in the model. The definitions are not unique, but offer an idea of how well the model can learn to approximate a wide variety of functions.

### 5.2 Performance as a function of complexity

Can model performance be predicted by the complexity of the model? Two examples are examined here using the models from the previous section. In the first example, an NN and a three-dimensional SISSO model are fitted to the DFT data used in ref. 19 using 80% of the data for training and 20% for testing. The SISSO model outperforms the best-performing NN architecture with a test root-mean-squared-logarithmic error of 0.231 compared to 1.367, despite having fewer parameters. The NN complexity can be scaled by changing the number of nodes and layers in the model. An NN with the same number of linear, nonlinear, and constant parameters as the SISSO model still does an order of magnitude worse than the SISSO model. This may not come as a surprise, since NNs are considered to be data hungry.[28] From this example, however, we can deduce that the number of constant, linear, and nonlinear parameters in the model is not enough information to predict model performance. Rather, the model class is the determining factor here.

In the second example, an NN and an RF are trained to predict bandgaps on the AFLOW dataset of ref. 8. Features describing the elemental composition of the materials, as described in ref. 29, are fed into both models. For the same parameter count, one can argue that an NN with the ReLU activation function is more complex than the RF, since the RF model is piecewise constant and the NN is piecewise linear. However, both NNs and RFs are universal approximators, meaning that with enough splits/nodes, they can approximate any probability distribution. On the AFLOW bandgap dataset, after cross-validation, the RF outperforms the NN, with test MAEs of 469 meV and 515 meV, respectively. This trend holds when we restrict the total number of parameters of both models to be similar. Once again, model class, not model complexity, is the decisive factor of model performance.

We can conclude from both examples that the number of trainable parameters alone tells us very little about model performance. Rather, we find that for a given dataset, the performance depends on the model class.

### 5.3 Searching for a definition of model complexity

Can model complexity be defined with a scalar? In the previous subsections, we saw that the total parameter count is a poor metric for NNs. A similar argument can be made for polynomial regressors, where some parameters allow for linear terms to be used by the model and others allow for nonlinear terms. One could therefore be motivated to count the coefficients to linear and nonlinear terms and place them on uneven footing, since the nonlinear coefficients give the model a different type of flexibility than the linear coefficients. The question though is, how to weight these terms to come up with a more general metric for model complexity. The simplest combination would be a weighted sum:

$$C(\mathrm{h}) = a \times A(\mathrm{h}) + b \times L(\mathrm{h}) + c \times N(\mathrm{h}) \tag{4}$$

where $C$ is the complexity of the model h, $A$ is the number of additive parameters, $L$ is the number of multiplicative coefficients to linear terms in the features, and $N$ is the number of coefficients of nonlinear combinations. If we set the coefficients of the complexity metric, $a$, $b$, and $c$, to unity, we recover the total trainable parameter count. Assigning values to $a$, $b$, and $c$ is, however, not an easy task. The

proposed complexity metric will mean different things for different models. As discussed, RFs have additive constant parameters that need to be learned, but no coefficients to linear or nonlinear combinations of the features. So with 10 000 additive constant parameters, the RF can learn to approximate many different distributions with a low mean-squared error. A generalized linear model, however, is only able to approximate constant functions with constant additive parameters. Therefore, with the same value of complexity, as defined in eqn (4), the RF has a much higher capacity to learn a wide variety of distributions than the generalized linear model.

These examples demonstrate that the total parameter count is not sufficiently descriptive of the model's ability to learn to approximate a wide variety of functions for many model classes. We offer an alternative metric for model complexity as a weighted sum of constant, linear, and nonlinear terms. We conclude, however, that the meaning of complexity is model dependent. This motivates us to therefore only talk about complexity within a single model class. More work is needed to explore these topics more carefully from an information theoretic point of view.

# 6 Infrastructure requirements

## 6.1 File and storage requirements for dataset calculation

In order to create large curated datasets of DFT calculations, sophisticated high-throughput workflows have become an indispensable tool. Here, we would like to provide an estimate for what this means in terms of infrastructure requirements. In Fig. 6, an example of such a workflow is depicted, leaving out workflow managers,[30] validity constraints,[31] and the like for the sake of simplicity. Here, an Atomic Simulation Environment (ASE)[32] database is employed to store input crystal structures that the user desires to simulate with `exciting`, an all-electron code based on the linearized augmented planewave method.[33] In this specific

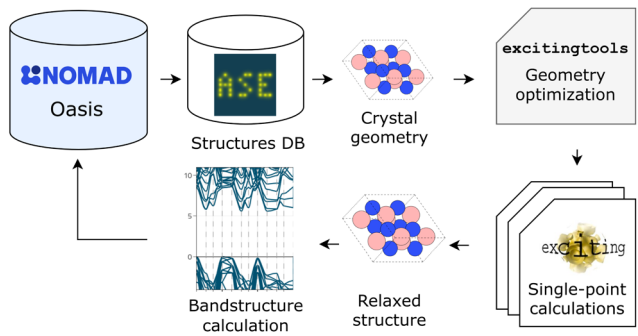


Fig. 6 Workflow for high-throughput geometry optimization, followed by a band-structure calculation. Crystal structures are pulled from a NOMAD Oasis instance and stored in an ASE database (DB). The structures are read by the Python API `excitingtools`, and a geometry optimization is carried out, consisting of multiple single-point ground-state calculations with `exciting`. For the resulting relaxed geometry, a bandstructure calculation is performed. All output files are uploaded to a NOMAD Oasis instance.

workflow, each structure is relaxed, *i.e.*, the geometry is optimized by running multiple single-point calculations, *via* the open-source Python API `excitingtools`.[34] The bandstructure is calculated afterwards for the equilibrium geometry. This rather simple workflow creates 41 output files and roughly 30 MB of data for each crystal structure. In addition, for each structure, the calculation must be converged with respect to BZ sampling and basis-set size. Typical usage would see at least three different $k$-point grids and three different basis-set sizes. For 30 000 crystal structures, which is not too big a number for ML tasks, (see, *e.g.*, Section 2), and nine different settings per structure, we perform 270 000 relaxations and bandstructure calculations. This amounts to 11 070 000 files, or in terms of storage, roughly 8.1 TB of data.

Such workflows are typically executed on supercomputers, which impose limits on users concerning disk space and numbers of files. Storage limits are, *e.g.*, on the order of a few 100 GB in backed-up directories, or a few TB in scratch directories. There are also limits in terms of the number of files that can be stored (often at the most 1 million). Both limits often mean that the users need to compress their data periodically and transfer their files to a storage system outside of the supercomputer network. In this example, the data are transferred to a NOMAD Oasis.[35,36] NOMAD Oasis is the same software as in the public NOMAD data infrastructure,[37] including, among other tools, parsing, normalizing, electronic notebooks, and the NOMAD AI toolkit.[38] The software can be installed locally by any research group and can be configured to their needs. Calculated data, as described above, are stored into an interoperable format.[39,40]

In summary, calculating big data presents its own unique infrastructure challenges. The workflow example shown here quantitatively demonstrates that performing high-throughput DFT calculations requires sophisticated hardware/software solutions to navigate HPC file and storage restrictions. The NOMAD Oasis is one such solution to this problem, which also allows for making data ready for ML purposes.

### 6.2 Computing requirements for training large models

The state of the art in ML is to employ meta-learning approaches, or neural architecture searches (NAS) to find the best NN model architecture.[41] Typically 500 to 10 000 architectures are selected by an algorithm (*e.g.*, random search or reinforcement learning) and trained, and then the best ones are selected by a user-defined reward function.[42] This approach has been applied to a wide variety of fields from object detection[43] to audio-signal processing,[44] and, more recently, in computational materials science.[8] It can be formulated mathematically as:

$$\max_{h\in H, D\sim P(\mathbf{x})} \mathcal{R}(h(D)) \tag{5}$$

where the reward function $\mathcal{R}$ is maximized by searching for the NN architecture, $h$, in a user-specified search space, $H$, over some given dataset $D$. The dataset is defined as a collection of independently and identically distributed data points, *i.e.*, $D = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, (\mathbf{x}_n)\}$, where data points $\mathbf{x}_i$ are sampled from some underlying probability distribution $P(\mathbf{x})$. A simple search space may consist of the layers (*e.g.*, 1–5) in a deep neural network (DNN) and the nodes per layer (*e.g.*, one of [16, 32, 64]). The NAS needs to train each candidate architecture enough to be able to compare them against one another. Note that there are some methods (*e.g.*,

differentiable search,[45] shared weights,[43] early stopping,[44] *etc.*) in the literature to reduce the computational burden of NAS, but none are guaranteed to return the correct rankings of architectures in terms of the reward function. For an MPNN model, several architecture parameters determine the number of model parameters. Here we discuss two such parameters, the number of message-passing steps and the latent space size.[11] There are also other parameters, such as the readout function and the number of layers in each graph convolution, which we do not discuss here.

Fig. 7 shows how the performance of the MPNN architecture described in ref. 11 and trained on AFLOW bandgaps depends on these two parameters. Here, the NAS reward function is the negative of the validation RMSE. Each data point in the figure represents an individual MPNN architecture. The $z$-axis represents this architecture's best validation RMSE across different initial learning rates, learning decay rates, batch sizes, dropouts, and readout functions. We see that the validation RMSE as a function of the two architecture parameters is non-convex. This non-convexity is what makes the optimization of the architecture slow and why black-box optimization methods such as reinforcement learning, random search, or evolutionary search are necessary.[46] Whether the NAS reward function is non-convex depends, of course, on the dataset and the search space. The mean (median) validation RMSE of a NAS candidate architecture across the entire space, not just these two parameters, is 556 meV (553 meV). The best candidate, which has a latent size of 128 and uses 3 MP steps, has a validation RMSE of 474 meV. This means that the optimal NAS architecture yields an improvement of 80 meV over the mean architecture in our search space, or in relative terms, 14.7%. To give an example, for finding solar cell materials with a bandgap between 1.1 and 1.5 eV, an 80 meV improvement in the model prediction may not be vital but would certainly be desired.

Let us proceed to analyze the computing requirements for a concrete NAS example. The SchNet graph neural network has *ca.* 85 000 trainable parameters for a latent size of 64 and three message-passing steps (when trained on a dataset

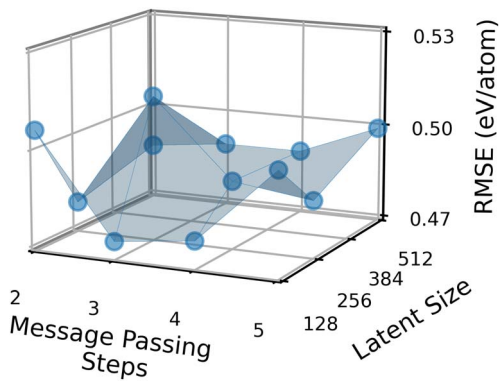


Fig. 7 Effect of two architecture parameters, number of message-passing steps and latent space size on the validation RMSE of a message-passing neural network with edge updates to fit bandgaps of the AFLOW dataset. Other parameters, such as the learning rate and size of the readout function, were optimized for each message-passing step and latent size combination.

with 81 different atomic elements).[47] On a single NVIDIA V100 GPU, it takes about two hours to train 2 000 000 steps on the AFLOW bandgap dataset (batch size 32). A further breakdown of the time required per training step shows that the model takes 0.0010 seconds on average to batch the data and 0.0023 seconds to perform a gradient-update step. For the SchNet-derived NAS architecture search space, approximately 2 000 000 steps are needed to have an idea of whether a candidate architecture is promising or not. This means that to train 2000 architectures, approximately 40 000 GPU hours for this dataset are needed, or in other words, more than 4.5 years on a single V100 GPU. Converting this number to dollars, current Amazon Web Services pricing plans fluctuate around 3 USD per NVIDIA V100 GPU hour.[48] Thus, a budget of roughly 120 000 USD is needed to perform this NAS. If we use a larger model than SchNet, such as PaiNN[49] (600 000 *vs.* 85 000 parameters), the training time and subsequent GPU and dollar budget would increase significantly. Moreover, considering larger datasets than AFLOW (about 60 000 data points, see Section 2), training would also slow down dramatically.

We can also compare CPU *versus* GPU training for this example. On a single CPU core, the training of 2 000 000 training steps of the SchNet model takes 10.5 hours, or on average approximately 0.0012 seconds each training step to batch the data, and 0.0177 seconds to take a gradient step and update the parameters. We conclude that CPU training is nearly an order of magnitude slower.

As NAS becomes more ubiquitous in materials science, as it has in other fields, new infrastructure challenges will have to be addressed. Finding the optimal model is often a non-convex task and can belong to large multidimensional search spaces that require black-box optimization methods. For models with a relatively small number of parameters, an MPNN NAS can require a huge computational and therefore monetary budget to be run on medium-sized to large datasets. As datasets and the base models being trained grow larger, so too will the infrastructure requirements.

# 7 Discussion and conclusions

If we define a dataset as "big" because it has a large number of data points, we may wonder why ML models fail to generalize. For this aspect, data diversity, *i.e.*, how well the underlying physical space is sampled, is critical. For example, AFLOW and MP datasets, despite their size, do not appear to contain enough data to describe the wide variety of materials. In other words, even larger and – in particular – more diverse datasets are required to build robust and transferable models. This has been shown by combining these two datasets.

Efforts to create big datasets will require advanced technical and software solutions. For example, computing only a fraction of the above mentioned datasets causes file and storage issues. In addition, as datasets grow, so does the number of model parameters. MPNN and other variants of graph neural networks have become quite common in materials science. Finding the best MPNN model requires a neural-architecture search over an often non-convex search space. GPUs offer an order of magnitude speedup in training, but using enough GPUs to perform a single NAS search over a large dataset requires a significant monetary budget.

Databases also pose challenges related to data veracity. For instance, data points computed for the same material may differ in the input settings. Sorting or clustering data using similarity metrics can help users better understand the

quality of data. It also enables ML practitioners to filter data into homogeneous subsets that contain less variation in the target properties. Multi-fidelity modeling,[13] where hetereogeneous data can be used by the model, is not discussed here, but would be an alternative option for dealing with data veracity.

Big data presents an opportunity to use complex models. Complexity is shown to be well defined within the confines of a single model type. A general definition for comparative purposes is, however, lacking. The total trainable parameter count, or the number of constant, linear, or nonlinear terms, can be less important than the model class when predicting performance. Increasing complexity, however, as shown with the example of using nonlinear features in cluster expansion, can aid significantly in describing intricate physics. More research is needed to better define model complexity in a data- and model-independent manner.

In this work, we attempt to shine light on some aspects of big data in materials science. There are certainly many more aspects to be explored. We hope that the issues we illustrate here will motivate further research along these lines.

## Data availability

The code used to gather and generate AFLOW dataset splits, perform neural architecture searches on MPNNs and profile the batching and gradient update steps of MPNNs can be found here: **https://github.com/tisabe/jraph_mpeu**. The SISSO and NN models fitted to ref. 19 can be found at this link: **https://nomad-lab.eu/aitutorials/error-estimates-qrf**. The NN and RF models fitted to the AFLOW bandgaps can be found here **https://colab.research.google.com/drive/1d52Z5QLC9qdU06xTa6X9c1UGEylcrkoq?usp=sharing**.

## Author contributions

C. D. sketched the idea and drafted the manuscript. T. B. and D. S. provided the example on model transferability; S. R. provided the cluster-expansion example. M. K. provided the example on similarity measures. D. S. and L. G. led the discussion on model complexity. D. S. and T. B. provided the example on infrastructure requirements. All authors contributed to the writing of the manuscript.

## Conflicts of interest

There are no conflicts to declare.

## Acknowledgements

Work was carried out in part by the Max Planck Graduate Center for Quantum Materials. Partial funding is appreciated from the German Science Foundation (DFG) through the CRC FONDA (project 414984028) and the NIFD consortium FAIRmat (project 460197019), and the European Union's Horizon 2020 research and innovation program under the grant agreement No. 951786 (NOMAD CoE). D. S. acknowledges support by the IMPRS for Elementary Processes in Physical Chemistry. Computing time for the `exciting` workflow example was granted by

the Resource Allocation Board and provided on the supercomputers Lise and Emmy at NHR@ZIB and NHR@Göttingen as part of the NHR infrastructure (bep00098). We thank Nakib Protik and Caja Annweiler for valuable feedback on the manuscript.



## Notes and references

1 C. Draxl and M. Scheffler, *Handbook of Materials Modeling: Methods: Theory and Modeling*, 2020, pp. 49–73.
2 D. T. Speckhard, C. Carbogno, L. Ghiringhelli, S. Lubeck, M. Scheffler and C. Draxl, *arXiv*, preprint, arXiv:2303.14760, 2023, DOI: **10.48550/arXiv.2303.14760**.
3 D. Jha, L. Ward, A. Paul, W.-K. Liao, A. Choudhary, C. Wolverton and A. Agrawal, *Sci. Rep.*, 2018, **8**, 17593.
4 A. Jain, S. P. Ong, G. Hautier, W. Chen, W. D. Richards, S. Dacek, S. Cholia, D. Gunter, D. Skinner, G. Ceder and K. A. Persson, *APL Mater.*, 2013, **1**, 011002.
5 C. E. Calderon, J. J. Plata, C. Toher, C. Oses, O. Levy, M. Fornari, A. Natan, M. J. Mehl, G. Hart, M. B. Nardelli, *et al.*, *Comput. Mater. Sci.*, 2015, **108**, 233–238.
6 G. Kresse and J. Furthmüller, *Phys. Rev. B: Condens. Matter Mater. Phys.*, 1996, **54**, 11169.
7 J. P. Perdew, K. Burke and M. Ernzerhof, *Phys. Rev. Lett.*, 1996, **77**, 3865.
8 T. Bechtel, D. T. Speckhard, J. Godwin and C. Draxl, *arXiv*, preprint, arXiv:2309.06348, 2023, DOI: **10.48550/arXiv.2309.06348**.
9 C. Chen, W. Ye, Y. Zuo, C. Zheng and S. P. Ong, *Chem. Mater.*, 2019, **31**, 3564–3572.
10 J. Schmidt, L. Pettersson, C. Verdozzi, S. Botti and M. A. Marques, *Sci. Adv.*, 2021, **7**, eabi7948.
11 P. B. Jørgensen, K. W. Jacobsen and M. N. Schmidt, *arXiv*, preprint, arXiv:1806.03146, 2018, DOI: **10.48550/arXiv.1806.03146**.
12 K. Lejaeghere, G. Bihlmayer, T. Björkman, P. Blaha, S. Blügel, V. Blum, D. Caliste, I. E. Castelli, S. J. Clark, A. Dal Corso, *et al.*, *Science*, 2016, **351**, aad3000.
13 B. Huang, N. O. Symonds and O. A. von Lilienfeld, *Handbook of Materials Modeling: Methods: Theory and Modeling*, 2020, pp. 1883–1909.
14 M. Kuban, Š. Gabaj, W. Aggoune, C. Vona, S. Rigamonti and C. Draxl, *MRS Bull.*, 2022, **47**, 991–999.
15 M. Kuban, S. Rigamonti and C. Draxl, *MADAS – A Python Framework for Assessing Similarity in Materials-Science Data*, 2024.
16 C. Draxl and M. Scheffler, *MRS Bull.*, 2018, **43**, 676–682.
17 NOMAD, Numerical Errors FHI-aims Dataset, 2022, DOI: **10.17172/NOMAD/2020.07.27-1**.
18 V. Blum, R. Gehrke, F. Hanke, P. Havu, V. Havu, X. Ren, K. Reuter and M. Scheffler, *Comput. Phys. Commun.*, 2009, **180**, 2175–2196.
19 C. Carbogno, K. S. Thygesen, B. Bieniek, C. Draxl, L. M. Ghiringhelli, A. Gulans, O. T. Hofmann, K. W. Jacobsen, S. Lubeck, J. J. Mortensen, *et al.*, *npj Comput. Mater.*, 2022, **8**, 69.
20 M. Kuban, S. Rigamonti, M. Scheidgen and C. Draxl, *Sci. Data*, 2022, **9**, 646.

21 P. Willett, J. M. Barnard and G. M. Downs, *J. Chem. Inf. Comput. Sci.*, 1998, **38**, 983–996.
22 J. Sanchez, F. Ducastelle and D. Gratias, *Phys. A*, 1984, **128**, 334–350.
23 S. Rigamonti, M. Troppenz, M. Kuban, A. Huebner and C. Draxl, *arXiv*, preprint, arXiv:2310.18223, 2023, DOI: **10.48550/arXiv.2310.18223**.
24 A. Stroth, C. Draxl, and S. Rigamonti, Cluster expansion toward nonlinear modeling and classification, 2024, submitted.
25 R. Ouyang, S. Curtarolo, E. Ahmetcik, M. Scheffler and L. M. Ghiringhelli, *Phys. Rev. Mater.*, 2018, **2**, 083802.
26 M. Bianchini and F. Scarselli, *IEEE Transact. Neural Networks Learn. Syst.*, 2014, **25**, 1553–1565.
27 T. A. R. Purcell, M. Scheffler and L. M. Ghiringhelli, *J. Chem. Phys.*, 2023, **159**, 114110.
28 C. C. Aggarwal, *et al.*, *Neural Networks and Deep Learning*, Springer, 2018, vol. 10.
29 L. Ward, A. Agrawal, A. Choudhary and C. Wolverton, *npj Comput. Mater.*, 2016, **2**, 1–7.
30 A. S. Rosen, M. Gallant, J. George, J. Riebesell, H. Sahasrabuddhe, J.-X. Shen, M. Wen, M. L. Evans, G. Petretto, D. Waroquiers, *et al.*, *J. Open Source Softw.*, 2024, **9**, 5995.
31 F. Schintke, K. Belhajjame, N. De Mecquenem, D. Frantz, V. E. Guarino, M. Hilbrich, F. Lehmann, P. Missier, R. Sattler, J. A. Sparka, *et al.*, *Future Generat. Comput. Syst.*, 2024, **157**, 82–97.
32 A. H. Larsen, J. J. Mortensen, J. Blomqvist, I. E. Castelli, R. Christensen, M. Dułak, J. Friis, M. N. Groves, B. Hammer, C. Hargus, *et al.*, *J. Phys.: Condens. Matter*, 2017, **29**, 273002.
33 A. Gulans, S. Kontur, C. Meisenbichler, D. Nabok, P. Pavone, S. Rigamonti, S. Sagmeister, U. Werner and C. Draxl, *J. Phys.: Condens. Matter*, 2014, **26**, 363202.
34 A. Buccheri, F. Peschel, B. Maurer, M. Voiculescu, D. T. Speckhard, H. Kleine, E. Stephan, M. Kuban and C. Draxl, *J. Open Source Softw.*, 2023, **8**, 5148.
35 NOMAD, *Nomad Oasis Webpage*, 2024, **https://nomad-lab.eu/nomad-lab/nomad-oasis.html**.
36 M. Scheffler, M. Aeschlimann, M. Albrecht, T. Bereau, H.-J. Bungartz, C. Felser, M. Greiner, A. Groaa, C. T. Koch, K. Kremer, *et al.*, *Nature*, 2022, **604**, 635–642.
37 M. Scheidgen, L. Himanen, A. N. Ladines, D. Sikter, M. Nakhaee, Á. Fekete, T. Chang, A. Golparvar, J. A. Márquez, S. Brockhauser, S. Brückner, L. M. Ghiringhelli, F. Dietrich, D. Lehmberg, T. Denell, A. Albino, H. Näsström, S. Shabih, F. Dobener, M. Kühbach, R. Mozumder, J. F. Rudzinski, N. Daelman, J. M. Pizarro, M. Kuban, C. Salazar, P. Ondračka, H.-J. Bungartz and C. Draxl, *J. Open Source Softw.*, 2023, **8**, 5388.
38 L. Sbailò, Á. Fekete, L. M. Ghiringhelli and M. Scheffler, *npj Comput. Mater.*, 2022, **8**, 250.
39 C. Draxl and M. Scheffler, *J. Phys.: Mater.*, 2019, **2**, 036001.
40 L. M. Ghiringhelli, C. Baldauf, T. Bereau, S. Brockhauser, C. Carbogno, J. Chamanara, S. Cozzini, S. Curtarolo, C. Draxl, S. Dwaraknath, *et al.*, *Sci. Data*, 2023, **10**, 626.
41 B. Zoph and Q. V. Le, *arXiv*, preprint, arXiv:1611.01578, 2016, DOI: **10.48550/arXiv.1611.01578**.

42 M. Tan, B. Chen, R. Pang, V. Vasudevan, M. Sandler, A. Howard and Q. V. Le, *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 2820–2828.
43 G. Bender, H. Liu, B. Chen, G. Chu, S. Cheng, P.-J. Kindermans and Q. V. Le, *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 14323–14332.
44 D. T. Speckhard, K. Misiunas, S. Perel, T. Zhu, S. Carlile and M. Slaney, *Neural Comput. Appl.*, 2023, **35**, 12133–12144.
45 H. Liu, K. Simonyan and Y. Yang, *arXiv*, preprint, arXiv:1806.09055, 2018, DOI: **10.48550/arXiv.1806.09055**.
46 E. Real, A. Aggarwal, Y. Huang and Q. V. Le, *Proceedings of the Aaai Conference on Artificial Intelligence*, 2019, pp. 4780–4789.
47 K. Schütt, P.-J. Kindermans, H. E. Sauceda Felix, S. Chmiela, A. Tkatchenko and K.-R. Müller, *Adv. Neural Inf. Process. Syst.*, 2017, **30**, 992–1002.
48 Amazon, *Amazon Web Services EC2 P3 Instances*, 2024, **https://aws.amazon.com/ec2/instance-types/p3/**.
49 K. Schütt, O. Unke and M. Gastegger, *International Conference on Machine Learning*, 2021, pp. 9377–9388.