Digital Discovery

ROYAL SOCIETY OF CHEMISTRY

PAPER

View Article Online
View Journal | View Issue

Cite this: *Digital Discovery*, 2024, **3**, 2448

MADAS: a Python framework for assessing similarity in materials-science data

Martin Kuban, * Santiago Rigamonti and Claudia Draxl

Computational materials science produces large quantities of data, both in terms of high-throughput calculations and individual studies. Extracting knowledge from this large and heterogeneous pool of data is challenging due to the wide variety of computational methods and approximations, resulting in significant veracity in the sheer amount of available data. One way of dealing with the problem is using similarity measures to group data, but also to understand where possible differences may come from. Here, we present `MADAS`, a Python framework for computing similarity relations between material properties. It can be used to automate the download of data from various sources, compute descriptors and similarities between materials, analyze the relationship between materials through their properties, and can incorporate a variety of existing machine learning methods. We explain the architecture of the package and demonstrate its power with representative examples.

Received 13th August 2024
Accepted 16th September 2024

DOI: 10.1039/d4dd00258j

rsc.li/digitaldiscovery

1 Introduction

The discovery of novel materials is a crucial aspect of technological progress. Therefore, much emphasis is placed on the in-depth characterization of materials as well as on the synthesis and prediction of new ones. As a result of such investigations, the community produces enormous amounts of data, including both experimental and computational results. In this context, the need to make data FAIR[1] (Findable, Accessible, Interoperable, Re-useable), has become evident, and large, publicly available databases have been created to store and retrieve these data. Such large data volumes come with new challenges, such as making data available and comprehensible for researchers from different communities.

At the same time, such data collections enable new types of analysis, employing data-analytics and machine-learning (ML) methods. Obviously, the more (reliable) data become available, the better the results are. Conversely, with increasing amount of data, quality control becomes a bottleneck. This problem seems to be less critical in large materials-science databases that contain results of high-throughput (HT) calculations, where a consistent set of parameters is used to simulate many – often several thousands – of materials. In such HT efforts, the proper execution and convergence of the result is usually controlled by workflow description languages, such as ASR[2] or Jobflow,[3] or workflow engines, such as Fireworks,[4] AiiDA,[5] or MyQueue.[6] However, data contained in different high-throughput databases may not be comparable because of different approximations and computational settings used in the respective calculations. The effects of these differences are subject to recent studies, comparing all-electron to pseudopotential density-functional-theory codes[7] by either using a dedicated benchmark dataset, or by comparing material properties contained in different databases directly. Here, we exemplify in Fig. 1 the differences arising from different numerical approaches with results for two NaCl structures from three different HT materials databases. All calculations have been carried out with the VASP[11] code, employing density-functional theory (DFT) in the generalized-gradient approximation (GGA), more specifically the PBE parametrization. All structures are relaxed in terms of their volumes and atomic positions. We have

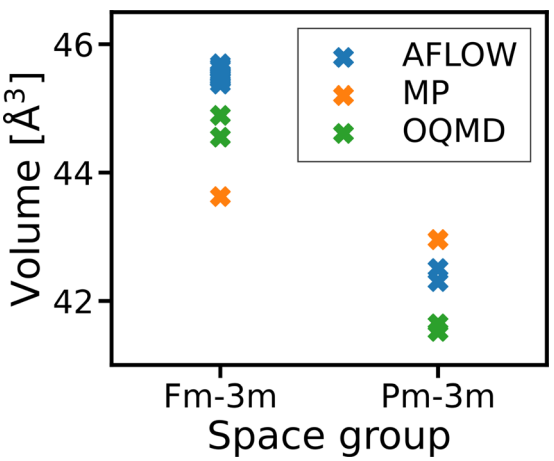

Fig. 1 Comparison of the unit-cell volume of NaCl obtained by DFT across the databases AFLOW,[8] Materials Project,[9] and OQMD.[10] The code to reproduce this analysis can be found in this publication's GitHub repository (https://github.com/kubanmar/madas-examples).

Institut für Physik and CSMB, Humboldt-Universität zu Berlin, Zum Großen Windkanal 2, 12489 Berlin, Germany. E-mail: kuban@physik.hu-berlin.de

verified that the structures are symmetrically equivalent by using the method described in ref. 12 and implemented in ASE,[13] version v3.22.1. Besides the same DFT approach in all three cases, the reported volumes differ by up to 2 Å^3. The results presented in ref. 14 suggest that these discrepancies are due to differences in the plane-wave cutoff and different relaxation schemes. One may argue that these errors are comparatively small, however, given the simple structures, they are still significant and may lead to differences in the corresponding properties. Since high-quality interoperable data are an important prerequisite for achieving high-precision predictions by ML models, combining these data for such a task can lead to significant uncertainties. To quantify the uncertainty in data, appropriate metrics need to be defined. Similarity measures can help to assess the quality of heterogeneous data.[15]

Similarity does not only concern data quality. It can be used to discover trends and outliers in consistent datasets[16,17] or visualize the contents of databases.[18] The concept of similarity holds, however, much greater potential, especially since large databases enable the search for materials with specific properties. For example, one might be interested in materials that are similar to each other in one aspect, but different in others. A prominent example is solar-cell materials that, unlike the most successful halide perovskites, should not contain lead. One challenge here is to develop a clear understanding of what we mean by similarity. When can we say that materials are similar? And how do we measure similarity?

During the last years, many descriptors for different aspects of materials have been published, focusing, for instance, on the atomic[19–22] or electronic[16,18,23] structure. Some of these descriptors, such as SOAP,[19] are specifically designed to measure the similarity between atomic configurations.[24] The available descriptors largely vary in complexity, ranging from a single number to high-dimensional representations that may demand significant computational resources. Despite the existence of comprehensive libraries such as `dscribe`[22] or `matminer`,[25] novel descriptors are usually published as stand-alone software packages. Integrating them in existing (or new) data-analysis workflows, requires 'glue code', that ensures the interoperability of the data formats required by the descriptors and other parts of the workflow. Such code tends to be application specific, not reusable, and hard to maintain.

In this work, we present `MADAS`, a Python framework that provides a modular, extendable, and simple interface to various tasks of similarity analysis of materials data. `MADAS` has been previously used in various illustrative examples, including the search for similar materials,[15] the analysis of the convergence behavior of the electronic structure in DFT calculations,[15] and the clustering of materials based on the similarity in terms of their electronic density of states.[16] The framework is equipped for various data-analysis tasks that use similarities with distinct but highly connected subtasks including (i) collection and storage of data from different sources, such as local file systems or remote databases, (ii) definition of material descriptors and similarity measures, (iii) calculation and storage of similarity relations, and (iv) analysis using a variety of techniques. In the following, we describe how we address related challenges by defining and implementing interfaces between the individual components of similarity analysis.

2 Results

An overview of the software architecture of `MADAS` is shown in Fig. 2. An API is used to download data from an external source, convert them to `Material` objects, and store them in a database. The file management of the database is handled by the backend. Data are retrieved from the database as `Material` objects and used to generate fingerprints, and similarity matrices are computed from them. Material properties, descriptors, and similarities can be combined for the use with AI tools. Data are exchanged between the components *via* a generic `Material` data class, which acts as a container for all data that can be used to calculate descriptors, a unique identifier (ID), and convenient methods to access these data. Below, we describe the core concepts that led to the development of `MADAS` and show different applications. Details on the implementation can be found in Section 3.

2.1 Collecting data from different sources

To make use of databases, *e.g.*, for a data-analysis or ML task, users can decide for a (open source) database and download the data they want to use *via* web APIs that are maintained by the database provider. However, despite efforts towards API standards,[26,27] a providers' proprietary API may give access to more specific information. It may also be subject to frequent changes. Therefore, to obtain and use the rich data provided by online repositories, users are often required to write custom scripts. These are not necessarily published together with the scientific results. Often, a (reproducible) description of how the data were obtained is omitted altogether.

These challenges must be addressed from a practical point of view: overall, users of online repositories have little influence on the design choices and availability of the data sources they rely on. This requires them to adapt their programs and workflows to any changes in online data. To efficiently work with the data under these conditions, programs must be easy to maintain and error-tolerant.

To support users in this respect, `MADAS` provides a Python class (see Section 3.1), which allows them to implement (and update) their own interfaces to external data (see Fig. 2, top left). By using the common naming schema and data class (*i.e.*, the `Material` class) for downloading data from different sources, neither the database, nor the data analysis pipeline (see Fig. 2) need to be changed when new data are added or their descriptions are changed by the providers of external databases. Thus, data analysis workflows can be re-used, speeding up their development.

2.2 Storing data locally

The data contained in online repositories are often subject to changes, for instance when new calculations are added or numerical parameters are refined. To ensure a persistent set of data to work on and to avoid repeated downloads from the same

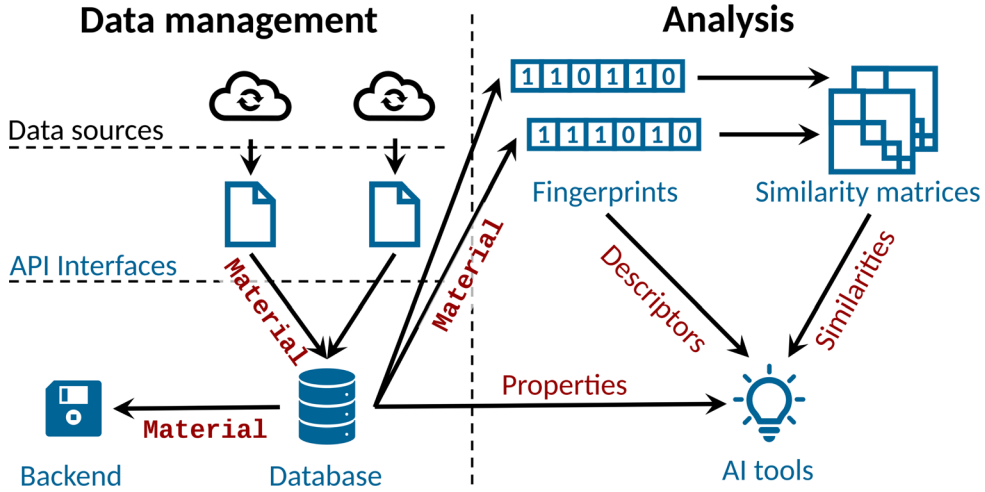

Fig. 2 Schematic workflow diagram for using the MADAS framework for data analysis. Symbols represent software components, arrows the data flow. Blue color indicates components that are explicitly implemented in MADAS, red labels annotate the type of data that is exchanged. The typewriter font of Material refers to the data class used within MADAS.

source, it is necessary to store the data locally. Here, this is realized by using a database. We note that, although the local storage of data can be realized by maintaining lists or binary object files, *e.g.*, supported by numpy[28] or pandas,[29] during the course of a research project, the number of generated lists can become large, and therefore hard to maintain and update. Thus, using a database brings several benefits. First, the data are stored in a consistent way. Therefore, unless altered on purpose, the original input data are preserved and the reproducibility of the analysis is strongly supported. Second, the consistent usage of unique identifiers (IDs) allows for connecting different parts of MADAS. For example, the results of clustering (see Section 2.5) based on the electronic structure of materials, can be related to the respective atomic structures contained in the database.

2.3 Fingerprinting materials

The next step after data collection, is their analysis (right panel of Fig. 2). This requires a suitable description of the data. One option is to extract properties as tabular data directly from the database (see Section 3.2). Alternatively, fingerprints can be used. In the context of this work, a fingerprint is the combination of a descriptor and a similarity measure. Descriptors (see also Section 1) are numerical representations of atomic configurations, and/or their respective properties. A similarity measure (sometimes also called a kernel) is a function S that maps any pair of descriptors (A, B) to a similarity score $0 \leq S(A, B) \leq 1$. A similarity score of $S = 1$ ($S = 0$) means that the descriptors are completely identical (different). The choice of the function S is arbitrary in general. However, depending on the problem that is addressed by fingerprinting materials, similarity measures with specific properties must be used. For most applications, symmetric measures, *i.e.*, $S(A, B) = S(B, A)$ are beneficial. For certain applications, such as clustering, a similarity measure whose complement $(1 - S)$ fulfills metric properties[30] can be necessary (see, *e.g.*, ref. 16). The verification of the former can be done analytically. In addition, MADAS provides a tool for verification of the metric properties for a given set of fingerprints, available as a Python class (madas.analysis.MetricSpaceTest). We note that a large number of similarity measures that can be used are directly available in scikit-learn.[31] Also, any metric $d(A, B) \in [0, \infty)$ that assigns a distance to a pair of fingerprints (A, B) can be transformed into a similarity measure S with $S(A, B) = 1/(1 + d(A, B))$.

An important distinction is to be made between fingerprint types and parameterizations. Fingerprints of the same type use the same descriptor. Many implementations of descriptors, however, depend on specific choices of parameters, *e.g.*, a cutoff or the number of basis functions.[19,20] Descriptors obtained with different parameters cannot be compared in a meaningful way. To avoid this situation, fingerprints of the same type can be distinguished in MADAS by their name, which is an identifier that is representative of the parameterization. When calculating the similarity between fingerprints, our implementation checks that both the type and the name are the same for the fingerprints to be compared, such to ensure that the computation of similarity is meaningful.

Our framework fosters rapid development of new fingerprints, since they can be tested on real data, and the computed descriptors can be passed to the analysis pipeline to analyze the results. It also supports code reuse, since new fingerprints can be published as (small) scripts and imported into different applications. For further information, including how to generate custom fingerprints, we refer to the documentation (**https://madas.readthedocs.io**).

The modular structure of MADAS allows for seamlessly calculating fingerprints that can be used to study different aspects of the data. To exemplify this, we use data from the C2DB[32,33] and visualize in Fig. 3 the materials most similar to $ZrTe_2$ using fingerprints of the electronic density-of-states (DOS) (see Section 3.3) with different parameterizations. Here, the DOS is discretized on a grid and represented as a binary-valued

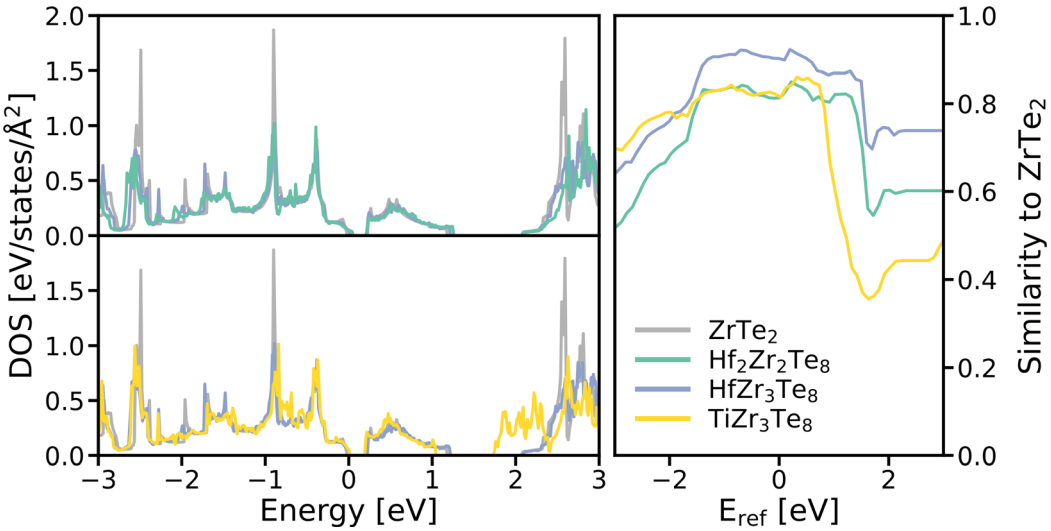

Fig. 3 Materials most similar to $ZrTe_2$, according to spectral fingerprints with different parameterizations (see main text): the top (bottom) left panel shows the DOS of the most similar materials when considering the conduction (valence) bands as the feature region. The right panel shows the energy-resolved similarity in a region around the band gap.

vector. The grid can be varied by various parameters to emphasize an energy region considered most relevant when comparing different spectra, *i.e.*, the feature region. These parameters include the reference energy E_{ref} (*i.e.*, the location of the grid's center) and the width w of the feature region, as well as an energy cutoff that controls the total width of the fingerprint. In this example, we show how the results depend on the chosen energy range: The most similar material to $ZrTe_2$ is $HfZr_3Te_8$, irrespective of whether we put the focus on the valence bands ($E_{ref} = -2$ eV, $w = 4$ eV), or on the conduction bands ($E_{ref} = 2$ eV, $w = 4$ eV), resulting in similarity values of $S = 0.83$ in both cases. The next most similar ones are $Hf_2Zr_2Te_8$ (top left panel) and $TiZr_3Te_8$ (bottom left panel) for both choices of the energy range, with a similarity of $S = 0.75$ and $S = 0.76$, respectively. To demonstrate which energy regions have the highest impact on the similarity, we compute spectral fingerprints in narrower energy windows of $w = 2$ eV, centered at a range of different reference energies, *i.e.*, between -3 and 3 eV. We then calculate the similarities between $ZrTe_2$ and its most similar materials for each of the reference energies. The result is shown in the right panel of Fig. 3. $HfZr_3Te_8$ is most similar to $ZrTe_2$ over almost the whole energy range with the exception of the lower valence bands ($E_{ref} < -1$ eV), where $TiZr_3Te_8$ has a higher similarity score. In the upper valence bands ($E_{ref} > 1$ eV), $Hf_2Zr_2Te_8$ has a higher similarity score than the former. Around the Fermi energy (-1 eV $\leq E \leq 1$ eV), their similarities to the reference material are almost identical.

This kind of analysis, which generates machine-readable output, can be used to quantify differences in scientific results, improving trust in data through quantitative analysis.

2.4 Similarity matrix

Given a set of fingerprints, the similarity relations between them can be calculated. The similarity matrix contains all pairwise similarities between members of a dataset. With a symmetric similarity measure, *i.e.*, $S(A, B) = S(B, A)$ for any two fingerprints A and B, the matrix is obviously symmetric.

Fig. 4 shows a similarity matrix of the DOS of $AlGaO_3$, obtained with the DFT code FHI-aims. These calculations are part of a dataset used to study the numerical quality of DFT calculations.[34] The subset used here has identical unit-cell volumes, while several other parameters vary, such as the number of k-points used for Brillouin zone sampling, the basis set size, the relativistic treatment, and the exchange–correlation functional. In the figure, we sort the matrix rows and columns by their mean similarity to the rest of the dataset, *i.e.*, the calculation that is on average most similar to all other calculations has the highest calculation index i_{calc}. Below the matrix, we show the number of k-points (blue) and the number of basis functions per atom (orange). The matrix exhibits a clear block structure, *i.e.*, there are subsets of calculations, whose members are more similar to other members of the set than to other calculations. There is a clear correlation between the computational parameters and the average similarity to the rest of the dataset. The calculations with lowest indices ($i_{calc} \leq 23$) are especially dissimilar, with average similarities $\bar{S} \leq 0.5$. We traced this back to artifacts in the DOS which appear when the scalar ZORA approximation is used for the relativistic treatment of core electrons in combination with too few k-points. When more k-points are used, these artifacts disappear. The next block in the matrix consists of calculations which have different combinations of low numbers of k-points and/or basis functions. The last block ($i_{calc} > 71$) with high average similarity has both sufficient k-points and basis-set sizes.

This kind of analysis can be automated and run as a workflow, speeding up convergence tests and allowing for automated discovery of reliable input parameters for DFT calculations. At the same time, it is highly interpretable and easily verifiable by humans, thereby increasing trust in the results. We note in passing that analyzing similarity measures in the form

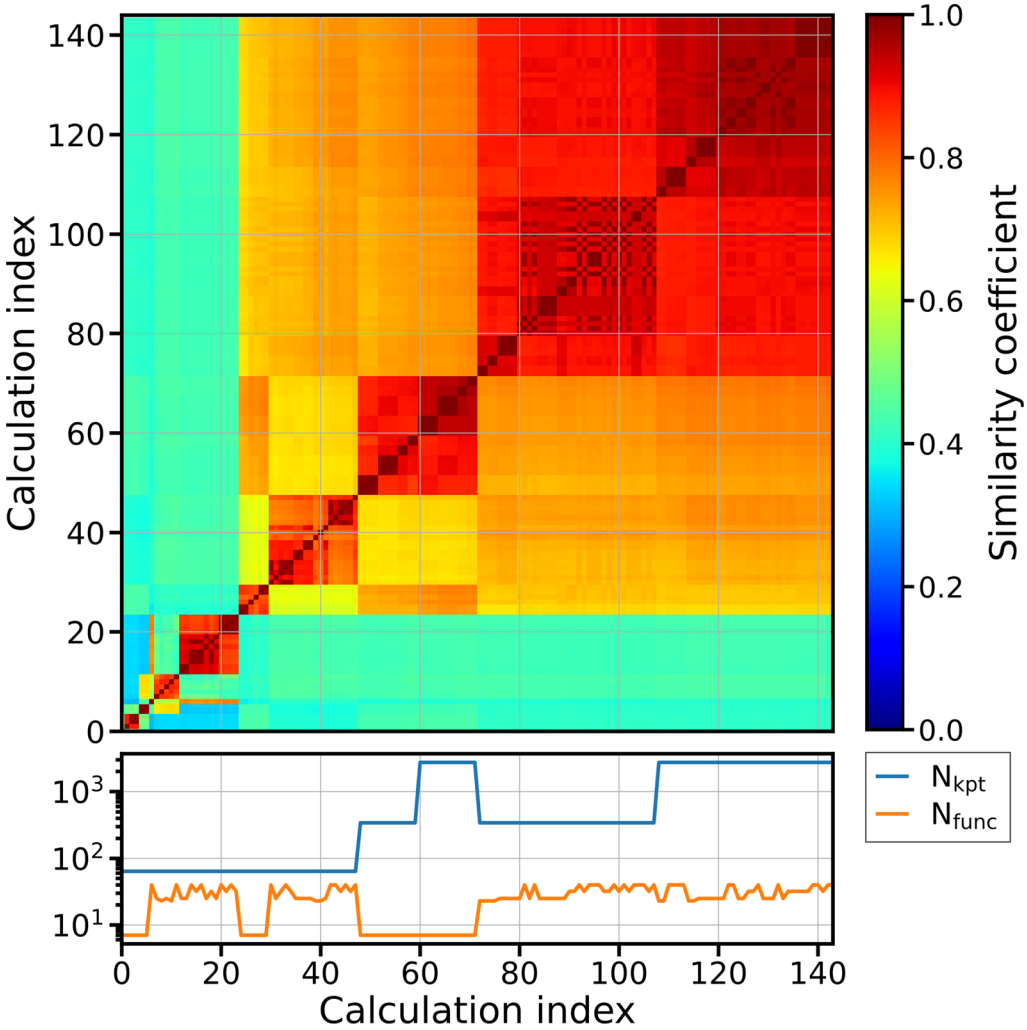

Fig. 4 Similarity matrix of the DOS of $AlGaO_3$ from data obtained with different basis-set sizes and k-point sets. The calculation with the highest average similarity to the rest of the dataset has the highest index. The bottom panel displays the number of k-points, N_{kpt}, and the number of basis functions, N_{func}, that were used for the ground-state calculations. For the calculation of the DOS, 9 times more k-points were used. The color code indicates the similarity coefficient, ranging from 0 (dark blue) to 1 (red).

illustrated here is generally only meaningful if a sufficiently large number of calculations are available.

For another example of a use case of our implementation, we refer to the NOMAD Encyclopedia[35] (**https://nomad-lab.eu/prod/rae/encyclopedia**), which features a list of materials with the most similar DOS for the majority of its entries. To obtain this information, we computed the full similarity matrix for all ~1.8 million materials for which a DOS was available. This was only feasible by massive parallelization, as supported by `MADAS`.

2.5 AI tools

`MADAS` can be used with a variety of artificial intelligence (AI) tools, including supervised and unsupervised learning. To achieve this, the functions and methods defined in `MADAS` are designed to be compatible with the API of `scikit-learn`.[31] For example, to find sets of materials that are similar to each other, a similarity matrix can be used as input for clustering algorithms.[16] Different to similarity searches, *i.e.*, finding the most similar materials for a single reference (see Section 2.3), clustering, as an unsupervised learning task, reveals global features of the dataset, by finding all sets of similar materials simultaneously.

In the following, we show how clustering can be used to analyze correlations between material properties. For this example, we have downloaded the crystal structures and electronic DOS for a dataset of 3847 cubic perovskites, which stem from the AFLOW database[8] and are also accessible through NOMAD.[36] We subsequently calculate PTE, DOS, and SOAP fingerprints (see Section 3.3) and the respective similarity matrices. The special version of SOAP fingerprints we use here reflects the atomic structure, but does not distinguish between atomic species. See Section 3.5 for details on how they are generated.

Fig. 5 shows the similarity matrices for the PTE (left column), SOAP (middle column), and DOS (right column) fingerprints, sorted by the results of the clustering process on the PTE matrix

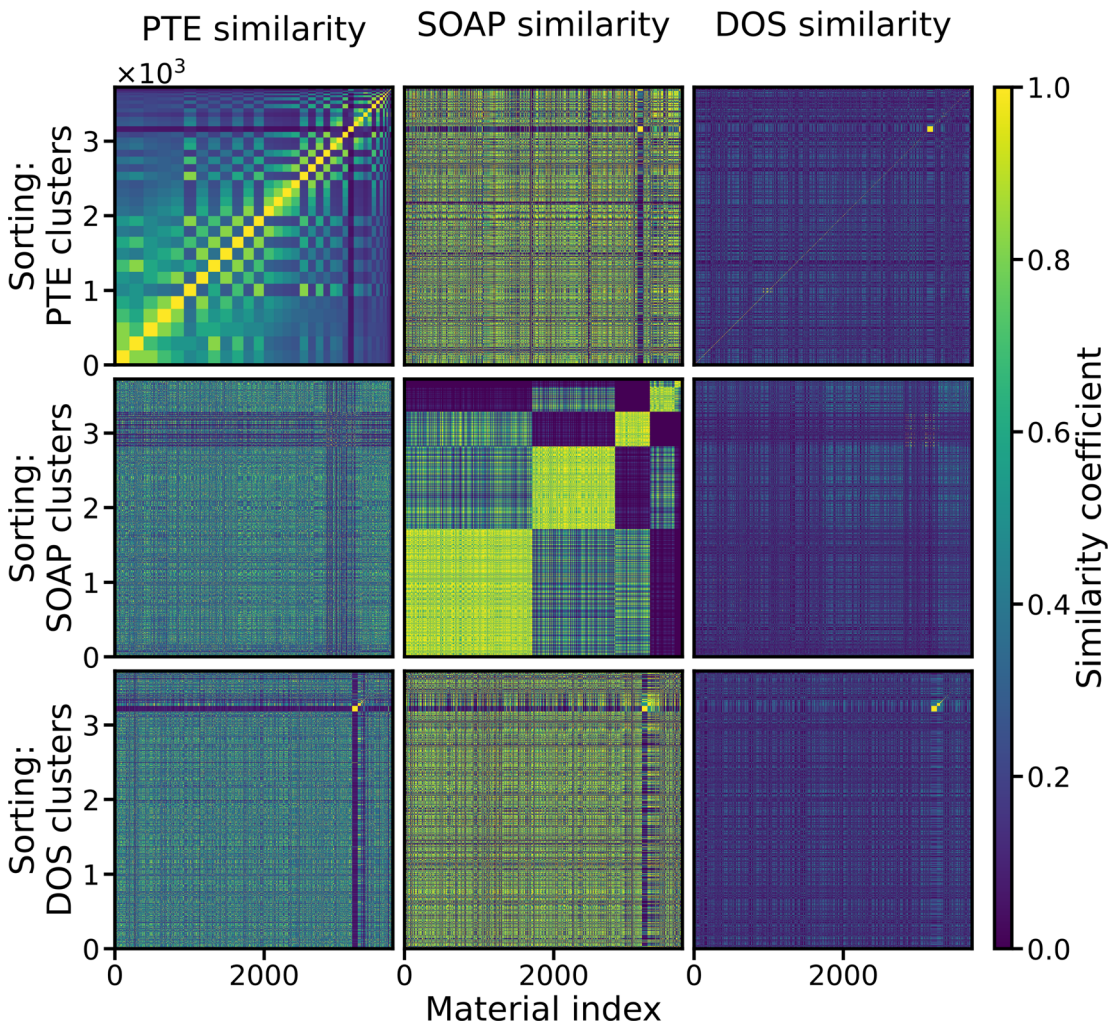

Fig. 5 Similarity matrices computed for ~3800 cubic perovskites. The columns of the grid correspond to the PTE (left), SOAP (middle), and DOS (right) fingerprints. In the top (middle, bottom) row of the grid, the materials in the similarity matrices are sorted such that they represent clusters that are found using the PTE (SOAP, DOS) similarity. The similarity coefficient is color coded, ranging from 0 (dark blue) to 1 (yellow).

(top row), the SOAP matrix (center row), and the DOS matrix (bottom row). The PTE matrix sorted by PTE clusters (top left panel) shows a many clusters of similar size, indicating a homogeneous distribution of elements across the materials in this dataset. Given the high-throughput, combinatorial approach of the AFLOWlib database, this was to be expected. The DOS matrix sorted by DOS clusters (bottom right panel) shows a different picture: The majority of the materials are outliers (indices <3182), *i.e.*, they don't have a similarity higher than $S = 0.75$ to any other material. At higher indices, all materials are contained in clusters, as defined by our implementation. However, we find only few large clusters, demonstrating the chemical diversity in this dataset. The SOAP matrix sorted by SOAP clusters (middle panel) shows 10 large clusters of different sizes, where the largest one has 1716 members. Since the descriptor that we use in this example does not distinguish between atomic species and the data set consists only of cubic perovskites, the cluster formation is likely to be related to the cell volumes.

Even more interesting are the off-diagonal elements of Fig. 5: Looking at the PTE similarity matrix sorted by DOS clusters (bottom left), the we see that the largest cluster, which contains 75 materials (indices 3182 to 3257), is also visible in the PTE matrix. Moreover, we also find it in the SOAP matrix. Upon closer inspection, it turns out that these are all calculations of $BPBa_3$.[37] This demonstrates the usefulness of our approach for detecting duplicates. The DOS similarity matrix sorted by PTE clusters (top right) reveals a slight correlation with the PTE descriptor, *i.e.*, a block-like structure appearing in the DOS similarity matrix. One might argue that materials with the same composition, and thus PTE descriptor, are statistically more likely to have a similar DOS. However, similarity in the electronic structure depends on many different factors, so the PTE descriptor certainly does not contain enough information to explain the DOS similarity. In the DOS matrix sorted by SOAP clusters (middle right), we see that the largest SOAP cluster shows no correlation with the DOS. However, the smaller clusters (index >1717) show a slightly higher than average similarity. Aside from the duplicate entries discussed above, the PTE and

SOAP matrices do not appear to be correlated. This aligns with our expectations, because they are agnostic of each other by construction.

The above analysis can be performed using MADAS in a few lines of code and can be found in the GitHub repository accompanying this paper (**https://github.com/kubanmar/madas-examples**). Moreover, to perform a similar analysis on another dataset, only the data query needs to be modified, resulting in a flexible and reusable workflow. We note that the analysis in ref. 16, also employed the unsupervised learning interface of MADAS.

We also note that supervised learning can be performed either by directly extracting descriptors from fingerprints or by using similarity matrices (see Section 2.4) as input to kernel-based machine-learning algorithms. The respective learning targets can be obtained from the database (see Section 3.2).

3 Implementation

3.1 Linking to external APIs

The interfaces to the APIs of online databases introduced in Section 2.1 are implemented in MADAS *via* a APIClass base class. Equipped with a common naming schema and data model, it can be used as a standardized template. That way, additional APIs can be added quickly. This template was used to write the code for generating Fig. 1, where minimal versions of API connections to the AFLOW,[8] Materials Project,[9] and OQMD[10] databases are utilized. Furthermore, MADAS implements convenient error mitigation and logging mechanisms. It natively supports an interface to the NOMAD Archive,[36] built on top of the NOMAD web API[35] and an interface to read local files.

3.2 Database

All methods related to the database, as described in Section 2.2, are implemented in MADAS in the MaterialsDatabase class. It ensures uniqueness of the data w.r.t. unique identifiers (IDs), so-called mids. With this persistent identification, no data are queried or stored twice, and the data entries can be linked with their original source. Properties stored in the database can be retrieved as tabular data *via* the get_property_dataframe method, which returns the data as a pandas[29] DataFrame object. The actual storage of data is handled by a backend, implemented in a Backend class, which is responsible for maintaining a connection to the database file, *i.e.*, for reading, writing, and updating its entries. By default, the AtomsDatabase of ASE[13] is used by the Backend, which implements an SQL schema for materials-science data. This enhances interoperability with other packages and allows for easy sharing of data. Different backends can be realized by writing custom Backend child classes that inherit its basic functionality. More details on how to achieve this, and tutorials for the implementation, can be found in the documentation (**https://madas.readthedocs.io**).

3.3 Fingerprints

All fingerprints, as introduced in Section 2.3, are implemented in MADAS *via* an extensible Fingerprint base-class, that is used consistently throughout all parts of the code. It allows for calculating the descriptors directly from database entries, *e.g.*, atomic structure and properties, and storing the results in the same database. The descriptors of the fingerprint can be retrieved using the Fingerprint().data attribute. Storing the fingerprints in the database is particularly useful for avoiding the repeated calculation of the descriptor in cases where this is resource intensive. The fingerprints can then be later reconstructed just from the database entries.

The fact that fingerprints are combinations of descriptors and similarity measures is reflected in the data model of MADAS, where each instance of a Fingerprint object is initialized with its respective similarity function. Then, the similarity between descriptors can be calculated by executing the get_similarity-method of one of the fingerprints. When another similarity metric is required, it can be changed by calling the set_similarity_function-method, without changing other parts of the program.

Currently, MADAS implements the spectral fingerprint used in ref. 15 and 16 (DOSFingerprint), the PTE (Peridoc Table of Elements) descriptor of ref. 16 (PTEFingerprint), a fingerprint for scalar properties (PROPFingerprint), and a test fingerprint (DUMMYFingerprint) for demonstration and testing purposes.

3.4 Similarity matrices

In MADAS, calculation, manipulation, and storage of similarity matrices (see Section 2.4) is implemented *via* a SimilarityMatrix class. A SimilarityMatrix object can be calculated from a list of Fingerprint (see Section 3.3) objects using its calculate method. The values stored in a SimilarityMatrix can be accessed by its matrix attribute.

Depending on the type of fingerprint that is used, calculating similarity matrices can be computationally demanding, due to the quadratic scaling, $\mathcal{O}(N^2)$, in the set size N. Therefore, we provide tools to optimize this task by considering unique entries, by parallelization, and/or by avoiding repetition of expensive calculations through storing results. Very large sets of fingerprints (about $\mathcal{O}(10^6)$) or computationally demanding similarity measures, may require to execute this task on a high-performance-computing (HPC) cluster. Since the entries of a similarity matrix are independent of each other, they can be computed in independent blocks. We provide an implementation of this functionality in a BatchedSimilarityMatrix class.

3.5 AI tools

The SOAP fingerprints used in Section 2.5 are generated using MADAS' interfaces to ASE and dscribe: we obtain the atomic structure from Material objects (see Section 2) as ASE Atoms objects and set all atomic species to the same element. Then we use the SOAP generator from dscribe to obtain the descriptor values, averaged over atom sites. As a similarity metric we use the pairwise Gaussian kernel of scikit-learn. We then cluster the similarity matrices using the threshold clustering method introduced in ref. 16. It can be used to find compact clusters where the similarity between cluster members is guaranteed to be larger than $2S_{\text{thres}} - 1$, where S_{thres} is the threshold used for

clustering. We used a threshold of $S_{\text{thres}} = 1$ for the PTE matrix, *i.e.*, all cluster members have an identical PTE descriptor, and a threshold of $S_{\text{thres}} = 0.75$ for the DOS and SOAP matrices.

For clustering similarity matrices, `MADAS` provides the wrapper class `SimilarityMatrixClusterer`. It simplifies the use of clustering algorithms, specifically, any clustering method that uses the naming conventions introduced in `scikit-learn`.

4 Discussion

With `MADAS`, we present a Python-based framework to support all steps of similarity analysis, including the collection and storage of data, the development and computation of fingerprints, the calculation of similarity matrices, and the integration of data-analytics methods. At the same time, `MADAS` is written in a modular way, allowing it to be customized to the application at hand, using only the parts of the code that are necessary for that particular task. The benefits of using `MADAS` lie not primarily in performance, but rather in flexibility and efficiency in prototyping and scripting (*via* an object model that favors reusability), error-tolerance through customizable exception handling, focus on data provenance through logging, and integration with well-established libraries. Individual components of `MADAS` can be easily integrated into existing data-analysis workflows *via* file exchange or *via* adapters (*e.g.*, `APIClass` or `Backend` classes, see Section 3) to allow for communication between workflow tasks.

We have demonstrated its use for managing data and comparing calculations across different external data sources, have shown how spectral fingerprints can be used and adapted to quantify local and global similarities of the electronic structure of materials, and have exemplified how similarity matrices can be used to group and rank calculations performed with different numerical settings.

Similarity searches are a well-known technique in molecular chemistry and drug discovery.[30,38] A common application is to scan a database of existing and hypothetical molecules for those that resemble a specific structural pattern that is assumed to correlate with favorable properties of a reference molecule. To do so, it is assumed that the presence or absence of molecular features, encoded in a fingerprint, correlates with the properties of the reference in terms of a so-called quantitative structure–property relationship (QSPR). That this is true in many cases is confirmed by the successes and continuous development of this technique. For materials, however, we find a different picture: The electronic structure (and therefore many derived properties of interest) is not necessarily determined by the local atomic structure, but reflects the intricate non-local many-body nature of extended systems. This situation asks for the development of novel, advanced fingerprints and techniques that can help to scan the materials space for interesting compounds. The spectral fingerprints highlighted here, should be just the beginning of such developments.

In the future, we will extend `MADAS` also with more tools to support (semi)automatic data analysis, outlier detection, and focus on data-quality assessment. The longevity of the code is supported by its rich documentation (**https://madas.readthedocs.io**) and modular design. This opens up development to the community and enables the code to grow with its user base.

Code availability

`MADAS` is released as open source under the Apache 2.0 license and available at the public Github repository **https://github.com/kubanmar/madas**. The code used is contained in release v1.0.5. The package is also available on the Python Package Index (PyPI) at **https://pypi.org/project/madas/**. The documentation source code can be found at **https://github.com/kubanmar/madas-docs**. All code required to reproduce the results in this manuscript is available at **https://github.com/kubanmar/madas-examples**.

Data availability

The identifier for the data stemming from NOMAD, AFLOW, the Materials Project, and OQMD, all accessed on August 13, 2024, used in Section 1, Section 2.4, and Section 2.5, respectively, can be found in the GitHub repository for this manuscript (**https://github.com/kubanmar/madas-examples**). The data used in Section 2.3 are available at **https://github.com/kubanmar/dos-fingerprints-data**.

Author contributions

M. K. wrote the code and documentation and analyzed the results. S. R. and C. D. contributed ideas and discussed and reviewed all parts of the work. All authors participated in the writing of the manuscript.

Conflicts of interest

There are no conflicts to declare.

Acknowledgements

This work received funding from the German Research Foundation (DFG) through the SFB 1404 (FONDA), project 414984028 and the NFDI consortium FAIRmat, project 460197019. We thank Šimon Gabaj for testing the code and Lauri Himanen and Nathan Daelman for valuable feedback to the manuscript.

Notes and references

1 M. D. Wilkinson, M. Dumontier, I. J. Aalbersberg, G. Appleton, M. Axton, A. Baak, N. Blomberg, J.-W. Boiten, L. B. da Silva Santos, P. E. Bourne, J. Bouwman, A. J. Brookes, T. Clark, M. Crosas, I. Dillo, O. Dumon, S. Edmunds, C. T. Evelo, R. Finkers, A. Gonzalez-Beltran, A. J. Gray, P. Groth, C. Goble, J. S. Grethe, J. Heringa, P. A. 't Hoen, R. Hooft, T. Kuhn, R. Kok, J. Kok, S. J. Lusher, M. E. Martone, A. Mons, A. L. Packer, B. Persson, P. Rocca-Serra, M. Roos, R. van Schaik, S.-A. Sansone, E. Schultes, T. Sengstag, T. Slater, G. Strawn,

M. A. Swertz, M. Thompson, J. van der Lei, E. van Mulligen, J. Velterop, A. Waagmeester, P. Wittenburg, K. Wolstencroft, J. Zhao and B. Mons, *Sci. Data*, 2016, **3**, 160018.

2 M. Gjerding, T. Skovhus, A. Rasmussen, F. Bertoldo, A. H. Larsen, J. J. Mortensen and K. S. Thygesen, *Comput. Mater. Sci.*, 2021, **199**, 110731.

3 A. S. Rosen, M. Gallant, J. George, J. Riebesell, H. Sahasrabuddhe, J.-X. Shen, M. Wen, M. L. Evans, G. Petretto, D. Waroquiers, G.-M. Rignanese, K. A. Persson, A. Jain and A. M. Ganose, *J. Open Source Softw.*, 2024, **9**, 5995.

4 A. Jain, S. P. Ong, W. Chen, B. Medasani, X. Qu, M. Kocher, M. Brafman, G. Petretto, G.-M. Rignanese, G. Hautier, D. Gunter and K. A. Persson, *Concurr. Comput. Pract. Exp.*, 2015, **27**, 5037–5059.

5 G. Pizzi, A. Cepellotti, R. Sabatini, N. Marzari and B. Kozinsky, *Comput. Mater. Sci.*, 2016, **111**, 218–230.

6 J. J. Mortensen, M. Gjerding and K. S. Thygesen, *J. Open Source Softw.*, 2020, **5**, 1844.

7 E. Bosoni, L. Beal, M. Bercx, P. Blaha, S. Blügel, J. Bröder, M. Callsen, S. Cottenier, A. Degomme, V. Dikan, K. Eimre, E. Flage-Larsen, M. Fornari, A. Garcia, L. Genovese, M. Giantomassi, S. P. Huber, H. Janssen, G. Kastlunger, M. Krack, G. Kresse, T. D. Kühne, K. Lejaeghere, G. K. H. Madsen, M. Marsman, N. Marzari, G. Michalicek, H. Mirhosseini, T. M. A. Müller, G. Petretto, C. J. Pickard, S. Poncé, G.-M. Rignanese, O. Rubel, T. Ruh, M. Sluydts, D. E. P. Vanpoucke, S. Vijay, M. Wolloch, D. Wortmann, A. V. Yakutovich, J. Yu, A. Zadoks, B. Zhu and G. Pizzi, *Nat. Rev. Phys.*, 2024, **6**, 45–58.

8 S. Curtarolo, W. Setyawan, S. Wang, J. Xue, K. Yang, R. H. Taylor, L. J. Nelson, G. L. Hart, S. Sanvito, M. Buongiorno-Nardelli, N. Mingo and O. Levy, *Comput. Mater. Sci.*, 2012, **58**, 227–235.

9 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.

10 S. Kirklin, J. E. Saal, B. Meredig, A. Thompson, J. W. Doak, M. Aykol, S. Rühl and C. Wolverton, *npj Comput. Mater.*, 2015, **1**, 15010.

11 G. Kresse and J. Furthmüller, *Phys. Rev. B: Condens. Matter Mater. Phys.*, 1996, **54**, 11169–11186.

12 D. C. Lonie and E. Zurek, *Comput. Phys. Commun.*, 2012, **183**, 690–697.

13 A. H. Larsen, J. J. Mortensen, J. Blomqvist, I. E. Castelli, R. Christensen, M. Dułak, J. Friis, M. N. Groves, B. Hammer, C. Hargus, E. D. Hermes, P. C. Jennings, P. B. Jensen, J. Kermode, J. R. Kitchin, E. L. Kolsbjerg, J. Kubal, K. Kaasbjerg, S. Lysgaard, J. B. Maronsson, T. Maxson, T. Olsen, L. Pastewka, A. Peterson, C. Rostgaard, J. Schiøtz, O. Schütt, M. Strange, K. S. Thygesen, T. Vegge, L. Vilhelmsen, M. Walter, Z. Zeng and K. W. Jacobsen, *J. Phys.: Condens. Matter*, 2017, **29**, 273002.

14 V. I. Hegde, C. K. H. Borg, Z. del Rosario, Y. Kim, M. Hutchinson, E. Antono, J. Ling, P. Saxe, J. E. Saal and B. Meredig, *Phys. Rev. Mater.*, 2023, **7**, 053805.

15 M. Kuban, Š. Gabaj, W. Aggoune, C. Vona, S. Rigamonti and C. Draxl, *MRS Bull.*, 2022, **47**, 991–999.

16 M. Kuban, S. Rigamonti, M. Scheidgen and C. Draxl, *Sci. Data*, 2022, **9**, 646.

17 E. Gazzarrini, R. K. Cersonsky, M. Bercx, C. S. Adorf and N. Marzari, *npj Comput. Mater.*, 2024, **10**, 73.

18 O. Isayev, D. Fourches, E. N. Muratov, C. Oses, K. Rasch, A. Tropsha and S. Curtarolo, *Chem. Mater.*, 2015, **27**, 735–743.

19 A. P. Bartók, R. Kondor and G. Csányi, *Phys. Rev. B: Condens. Matter Mater. Phys.*, 2013, **87**, 184115.

20 H. Huo and M. Rupp, *Mach Learn.-Sci. Techn.*, 2022, **3**, 045017.

21 M. F. Langer, A. Goeßmann and M. Rupp, *npj Comput. Mater.*, 2022, **8**, 41.

22 L. Himanen, M. O. Jäger, E. V. Morooka, F. Federici Canova, Y. S. Ranawat, D. Z. Gao, P. Rinke and A. S. Foster, *Comput. Phys. Commun.*, 2020, **247**, 106949.

23 N. Knøsgaard and K. Thygesen, *Nat. Commun.*, 2022, **13**, 468.

24 S. De, A. P. Bartók, G. Csányi and M. Ceriotti, *Phys. Chem. Chem. Phys.*, 2016, **18**, 13754–13769.

25 L. Ward, A. Dunn, A. Faghaninia, N. E. Zimmermann, S. Bajaj, Q. Wang, J. Montoya, J. Chen, K. Bystrom, M. Dylla, K. Chard, M. Asta, K. A. Persson, G. J. Snyder, I. Foster and A. Jain, *Comput. Mater. Sci.*, 2018, **152**, 60–69.

26 C. W. Andersen, R. Armiento, E. Blokhin, G. J. Conduit, S. Dwaraknath, M. L. Evans, Á. Fekete, A. Gopakumar, S. Gražulis, A. Merkys, F. Mohamed, C. Oses, G. Pizzi, G.-M. Rignanese, M. Scheidgen, L. Talirz, C. Toher, D. Winston, R. Aversa, K. Choudhary, P. Colinet, S. Curtarolo, D. Di Stefano, C. Draxl, S. Er, M. Esters, M. Fornari, M. Giantomassi, M. Govoni, G. Hautier, V. Hegde, M. K. Horton, P. Huck, G. Huhs, J. Hummelshøj, A. Kariryaa, B. Kozinsky, S. Kumbhar, M. Liu, N. Marzari, A. J. Morris, A. A. Mostofi, K. A. Persson, G. Petretto, T. Purcell, F. Ricci, F. Rose, M. Scheffler, D. Speckhard, M. Uhrin, A. Vaitkus, P. Villars, D. Waroquiers, C. Wolverton, M. Wu and X. Yang, *Sci. Data*, 2021, **8**, 217.

27 M. L. Evans, J. Bergsma, A. Merkys, C. W. Andersen, O. B. Andersson, D. Beltrán, E. Blokhin, T. M. Boland, R. Castañeda Balderas, K. Choudhary, A. Díaz Díaz, R. Domínguez García, H. Eckert, K. Eimre, M. E. Fuentes Montero, A. M. Krajewski, J. J. Mortensen, J. M. Nápoles Duarte, J. Pietryga, J. Qi, F. d. J. Trejo Carrillo, A. Vaitkus, J. Yu, A. Zettel, P. B. de Castro, J. Carlsson, T. F. T. Cerqueira, S. Divilov, H. Hajiyani, F. Hanke, K. Jose, C. Oses, J. Riebesell, J. Schmidt, D. Winston, C. Xie, X. Yang, S. Bonella, S. Botti, S. Curtarolo, C. Draxl, L. E. Fuentes Cobas, A. Hospital, Z.-K. Liu, M. A. L. Marques, N. Marzari, A. J. Morris, S. P. Ong, M. Orozco, K. A. Persson, K. S. Thygesen, C. Wolverton, M. Scheidgen, C. Toher, G. J. Conduit, G. Pizzi, S. Gražulis, G.-M. Rignanese and R. Armiento, *Digital Discovery*, 2024, **3**, 1509–1533.

28 C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van

Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke and T. E. Oliphant, *Nature*, 2020, **585**, 357–362.

29 W. McKinney, *Proceedings of the 9th Python in Science Conference*, 2010, pp. 56–61.

30 P. Willett, J. M. Barnard and G. M. Downs, *J. Chem. Inf. Comput. Sci.*, 1998, **38**, 983–996.

31 F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot and E. Duchesnay, *J. Mach. Learn. Res.*, 2011, **12**, 2825–2830.

32 S. Haastrup, M. Strange, M. Pandey, T. Deilmann, P. S. Schmidt, N. F. Hinsche, M. N. Gjerding, D. Torelli, P. M. Larsen, A. C. Riis-Jensen, J. Gath, K. W. Jacobsen, J. J. Mortensen, T. Olsen and K. S. Thygesen, *2D Mater.*, 2018, **5**, 042002.

33 M. N. Gjerding, A. Taghizadeh, A. Rasmussen, S. Ali, F. Bertoldo, T. Deilmann, N. R. Knøsgaard, M. Kruse, A. H. Larsen, S. Manti, T. G. Pedersen, U. Petralanda, T. Skovhus, M. K. Svendsen, J. J. Mortensen, T. Olsen and K. S. Thygesen, *2D Mater.*, 2021, **8**, 044002.

34 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, M. Strange, E. Wruss and M. Scheffler, *npj Comput. Mater.*, 2022, **8**, 69.

35 C. Draxl and M. Scheffler, *JPhys Mater.*, 2019, **2**, 036001.

36 C. Draxl and M. Scheffler, *MRS Bull.*, 2018, **43**, 676–682.

37 Searching the material on AFLOW (using the Aflux summons, **https://www.aflowlib.org/API/aflux/?compound,species(B,P,Ba),$nspecies(3),spacegroup_relax(221),$paging(1,1000)**) reveals $BPBa_3$ in space group number 221 has 66 unique entries in the database at the time of writing of this manuscript.

38 G. Maggiora, M. Vogt, D. Stumpfe and J. Bajorath, *J. Med. Chem.*, 2014, **57**, 3186–3204.