Computational Materials Science 247 (2025) 113495

Contents lists available at ScienceDirect

Computational Materials Science

journal homepage: www.elsevier.com/locate/commatsci

Full length article

Efficient structure-informed featurization and property prediction of ordered, dilute, and random atomic structures

Adam M. Krajewski [a,*], Jonathan W. Siegel [b], Zi-Kui Liu [a]

[a] Department of Materials Science and Engineering, The Pennsylvania State University, USA
[b] Department of Mathematics, Texas A&M University, USA

ARTICLE INFO

Keywords:
Atomistic
Machine learning
Structure-informed
Featurization
Throughput
Symmetry-equivalent
Voronoi
Efficient
Tessellation
Dilute
Defect
Solid solution
Python

ABSTRACT

Structure-informed materials informatics is a rapidly evolving discipline of materials science relying on the featurization of atomic structures or configurations to construct vector, voxel, graph, graphlet, and other representations useful for machine learning prediction of properties, fingerprinting, and generative design. This work discusses how current featurizers typically perform redundant calculations and how their efficiency could be improved by considering (1) fundamentals of crystallographic (orbits) equivalency to optimize ordered structures and (2) representation-dependent equivalency to optimize dilute, doped, and defect structures with broken symmetry. It also discusses and contrasts ways of (3) approximating random solid solutions occupying arbitrary lattices under such representations.

Efficiency improvements discussed in this work were implemented within `pySIPFENN` or *python toolset for Structure-Informed Property and Feature Engineering with Neural Networks* developed by authors since 2019 and shown to increase performance from 2 to 10 times for typical inputs. Throughout this work, the authors explicitly discuss how these advances can be applied to different kinds of similar tools in the community.

1. Introduction

`SIPFENN` or *Structure-Informed Prediction of Formation Energy using Neural Networks* software, first introduced by the authors in 2020 [1,2], is one of several open-source tools available in the literature [3–8,8,9] which train machine learning (ML) models on the data from large Density Functional Theory (DFT) based datasets like `OQMD` [10–12], `AFLOW` [13,14], Materials Project [15], NIST-`JARVIS`[16], `Alexandria` [17], or `GNoME` [18] to predict formation energies of arbitrary atomic structures, with accuracy high enough to act as a low-cost surrogate in the prediction of thermodynamic stability of ground and non-ground state configurations at 0K temperature. The low runtime cost allows such models to efficiently screen through millions of different atomic structures of interest on a personal machine in a reasonable time.

In addition to high-accuracy neural network models trained on `OQMD` [10–12], `SIPFENN` included a number of features not found in competing tools available at the time, such as the ability to quickly readjust models to a new chemical system based on just a few DFT data points through transfer learning and a selection of models optimized for different objectives like extrapolation to new materials instead of overfitting to high-data-density regions or low memory footprint [1].

`SIPFENN`'s usefulness has been demonstrated, for instance, in the cases where the structure of an experimentally observed compound could not be identified in industry-relevant Nd–Bi [19] and Al–Fe [20] systems and had to be predicted. This was accomplished by (1) high-throughput generation of hundreds of thousands of possible candidates with the exact stoichiometry based on elemental substitutions into structures from both open DFT-based databases [10–18] and experimentally observed ones from Crystallography Open Database (COD) [21–23], followed by (2) selection of thousands of low-energy candidates, (3) down-selection of tens of unique candidates based on clustering in the `SIPFENN`'s feature space, and (4) final validation with DFT and experiments. It has also been deployed in several thermodynamic modeling studies, e.g. of Nb–Ni system [24], in conjunction with DFT and experimental data processed through `ESPEI` [25] to automatically fit parameters of CALPHAD [26] models deployed in `pycalphad` [27].

* Corresponding author.
E-mail addresses: adam@phaseslab.org, ak@psu.edu (A.M. Krajewski).
URLs: https://github.com/amkrajewski (A.M. Krajewski), https://github.com/jwsiegel2510 (J.W. Siegel).

https://doi.org/10.1016/j.commatsci.2024.113495
Received 14 June 2024; Received in revised form 13 October 2024; Accepted 27 October 2024
Available online 7 November 2024

2. General structure featurization improvements

2.1. pySIPFENN overview and core advantages

Being able to predict the thermodynamic stability of arbitrary atomic structures and their modifications (e.g., formed by introduction of defects, strain, or selective displacements) is one of the most critical steps in establishing whether hypothetical candidates can be made in real life [28]; however, it is certainly not the only task of interest to the community. For instance, ML modeling of perovskites also covers properties like formability, bandgap, Curie temperature, Neel temperature, maximum magnetic entropy change, dielectric breakdown strength, and several other properties [29], while ML modeling of battery materials also covers properties such as redox potential, band gap, and dielectric breakdown [30]. These diverse needs, combined with increasing interest in multi-property modeling, have shifted the focus of `SIPFENN` tool from model training [1] toward the development of reliable, easy-to-use, and efficient general-purpose featurizers existing in a framework, which can be used by researchers and companies to quickly develop and deploy property-specific models, or use features directly in exploring similarity and trends in materials.

Thus, while the acronym has been retained, the name of the software has been changed to *python toolset for Structure-Informed Property and Feature Engineering with Neural Networks* or `pySIPFENN`, and the software component has been carefully re-implemented in its entirety to make it as general as possible and enable the following core advantages:

1. **Reliable featurization**, which can be immediately transferred to other tools thanks to standalone submodule implementations based only on two common libraries (`NumPy` [31] and `pymatgen` [32]). These include completely re-written `Ward2017` Java-based featurizer [3] (see Section 2.2) and 3 new ones, described in Sections 3, 4, and 5.
2. Effortless **plug-and-play deployment of neural network and other ML models**, for any property, utilizing any of the defined feature vectors, enabled by the use of Open Neural Network Exchange (`ONNX`) open-source format [33] which can be exported from nearly every modern ML framework and is then loaded into `pySIPFENN`'s `PyTorch` backend [34] through `onnx2torch` [35]. Furthermore, implementing custom predictors, beyond those supported by `PyTorch`, is made easy by design.
3. Dedicated `ModelExporters` submodule makes it **easy to export trained models for publication or deployment on a different device** while also enabling weight quantization and model graph optimizations to reduce memory requirements.
4. The ability to acquire or parse relevant data and adjust or completely retrain model weights through automated `ModelAdjusters` submodule, enabling several transfer learning schemes. **(a) Fine-tuning of models based on additional local data** to facilitate transfer learning ML schemes of the domain adaptation kind [36], where a model can be adjusted to new chemistry and specific calculation settings, introduced by `SIPFENN` back in 2020 [1], which is also being adopted by other toolsets like `ALIGNN` [37]. Such an approach can also be used iteratively in active learning schemes where new data is obtained and added. **(b) Tuning or retraining of the models based on community atomistic databases, or their subsets, accessed through OPTIMADE API** queried by `optimade-python-tools` library [38–40]. This allows for adjusting the model to a different domain, which in the context of DFT datasets could mean adjusting the model to predict properties with DFT settings used by that database or focusing its attention to specific chemistry like, for instance, all compounds of Sn and all perovskites. Critically, this functionality, in the default case (GGA+U formation energy from Materials Project [15]), only requires the user to provide a standard human-readable `OPTIMADE` query like the one below, while custom cases can be accomplished, bound mostly by the current limitations of OPTIMADE itself, as discussed in Appendix A.

```
'elements HAS "Hf" AND elements HAS "Mo"
AND NOT elements HAS ANY "O","C","F","Cl","S"'
```

The provenance of model tuning data can be always tracked by the data `names` list retained in the `OPTIMADEAdjuster` object, which are formed by reduced chemical and provider's data ID (e.g. `'HfTaTiB4MoC4-mp-1232383'` or `'HfZrNb2Mo-agm001451656'`), which can be used to look up complete data on providers website. For certain OPTIMADE providers, such as Alexandria [17], provenance is additionally exposed more directly in the form of `references` list of lists, which typically consist of related DOIs user can cite. **(c) Knowledge transfer learning** [41] to adjust models to entirely new, often less available properties while harvesting the existing pattern recognition.

The resulting `pySIPFENN` computational framework is composed of several components, as depicted in Fig. 1 serving as the schematic of both organization and data flow in a typical task. In a typical workflow, user interacts with the `core.Calculator` class, which acts as a container for ML models, orchestrates the tasks, and stores the results.

`pySIPFENN` is available through several means described in , alongside high-quality documentation and use tutorials. As depicted in Fig. 1, the individual sub-modules are designed to be relatively compartmentalized, allowing advanced users to quickly go beyond default workflow and implement custom solutions or utilize `pySIPFENN` components, such as structure featurizers, within another tool.

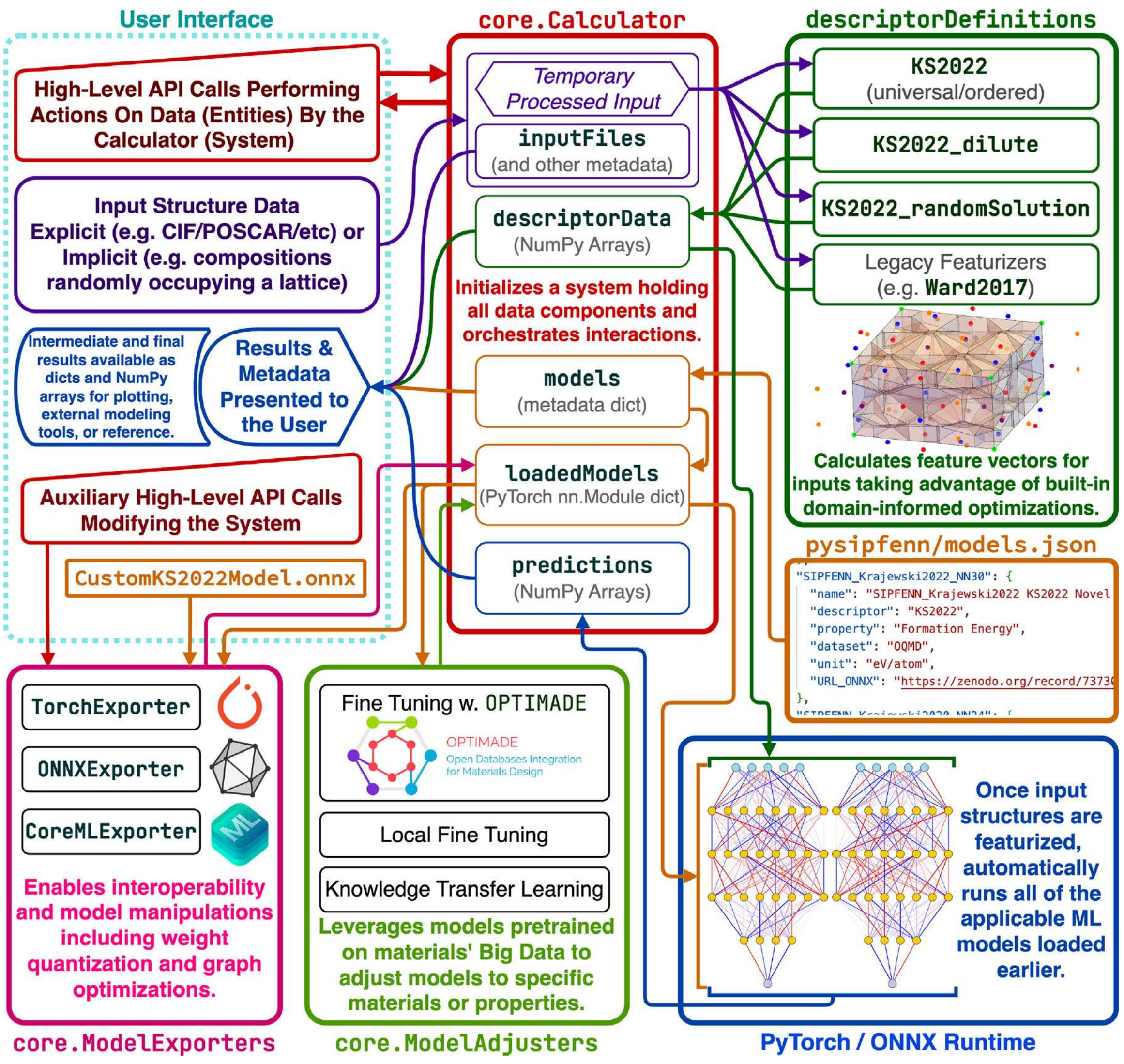
2.2. Ward2017 reimplementation

In their 2017 work Ward et al. [3] introduced a novel atomic structure featurization concept based on establishing and weighting neighbor interactions by faces from 3D Voronoi tessellation to describe local chemical environments (LCEs) of atomic sites and then performing statistics over them to obtain a global feature vector. The original `SIPFENN` models [2] built on top of this while utilizing an improved, carefully designed deep neural network models to obtain up to 2.5 times lower prediction error on the same dataset [1]. A detailed description of the descriptor can be found in Section 2.1 of Krajewski et al. [1]. In general, the calculation of the `Ward2017` descriptor covers three types of attributes:

- Based upon global averages over the components of the structure.
- Based upon local neighborhood averages for each site in the structure.
- More complex ones based upon averages over paths in the structure.

Ward et al. [3] implemented the above calculations in Java, which was popular at the time; while most of the current machine-learning packages use almost exclusively Python (e.g., `scikit-learn` [42] and `PyTorch` [34]), making it cumbersome to use Java. Even more critically, the original Java implementation was not computationally efficient (as explored in Sections 3, 4, and 5), and enabling tools were not supported in Java.

In the present work, authors have reimplemented Ward et al. [3] approach from scratch in Python as a standalone submodule for `pySIPFENN`, which calculates all 271 features within numerical precision, except for three based on a random walk on the structure(adjacent cells). The original implementation from Ward et al. [3] used a non-backtracking random walk, while this implementation switched to a backtracking random walk. Despite the exact reproduction of feature


User Interface
core.Calculator
descriptorDefinitions
High-Level API Calls Performing Actions On Data (Entities) By the Calculator (System)
Temporary Processed Input
inputFiles (and other metadata)
KS2022 (universal/ordered)
KS2022_dilute
KS2022_randomSolution
Legacy Featurizers (e.g. Ward2017)
Input Structure Data Explicit (e.g. CIF/POSCAR/etc) or Implicit (e.g. compositions randomly occupying a lattice)
descriptorData (NumPy Arrays)
Initializes a system holding all data components and orchestrates interactions.
Intermediate and final results available as dicts and NumPy arrays for plotting, external modeling tools, or reference.
Results & Metadata Presented to the User
models (metadata dict)
loadedModels (PyTorch nn.Module dict)
predictions (NumPy Arrays)
Calculates feature vectors for inputs taking advantage of built-in domain-informed optimizations.
Auxiliary High-Level API Calls Modifying the System
CustomKS2022Model.onnx
pysipfenn/models.json
TorchExporter
ONNXExporter
CoreMLExporter
Enables interoperability and model manipulations including weight quantization and graph optimizations.
core.ModelExporters
Fine Tuning w. OPTIMADE
OPTIMADE Open Databases Integration for Materials Design
Local Fine Tuning
Knowledge Transfer Learning
Leverages models pretrained on materials' Big Data to adjust models to specific materials or properties.
core.ModelAdjusters
Once input structures are featurized, automatically runs all of the applicable ML models loaded earlier.
PyTorch / ONNX Runtime

```
"SIPFENN_Krajewski2022_NN30": {
  "name": "SIPFENN_Krajewski2022 KS2022 Novel
  "descriptor": "KS2022",
  "property": "Formation Energy",
  "dataset": "OQMD",
  "unit": "eV/atom",
  "URL_ONNX": "https://zenodo.org/record/7373
},
```

Fig. 1. Main schematic of pySIPFENN framework detailing the interplay of internal components described in Section 2.1. The user interface provides a high-level API to process structural data within core.Calculator, pass it to featurization submodules in descriptorDefinitions to obtain vector representation, then passed to models defined in models.json and (typically) run automatically through all available models. All internal data of core.Calculator is accessible directly, enabling rapid customization. An auxiliary high-level API enables advanced users to operate and retrain the models.

values being desired, such an approach has been preferred due to more straightforward implementation and output stability against different seeds. The resulting difference in values of random walk features has been small enough to be ignored for practical purposes, being at most 10% compared to the reference values from Ward et al. [3], allowing ML models for formation energy to be deployed with the difference in predictions being, typically, 2–4 orders of magnitude lower than the reported model error. The Voronoi tessellation has been implemented with Voro++ [43–45] and all numerical operations were written using NumPy [31] arrays to greatly speed up the calculations and make the efficient utilization of different computing resources, such as GPUs, easy to implement.

2.3. KS2022 feature optimization

Typically, during feature engineering, researchers first attempt to collect all features expected to enable good inference and then remove some based on the interplay of several factors:

1. **Low impact on the performance**, since having more features increases the representation memory requirements and possibly increases the risk of overfitting to both systematic and random trends. The latter is very significant as it can lead to capturing noise instead of true patterns, excessive sensitivity to small input changes not expressed in the training domain, and poor generalization to new data/domains.
2. **High computational cost**, which limits the throughput of the method deployment.
3. **Unphysical features or feature representations** which can improve model performance against well-behaving benchmarks covering a small subset of the problem domain but compromise model interpretability and extrapolation ability in unpredictable ways.

The KS2022 feature set, added in pySIPFENN v0.10 in November 2022, is a significant modification of the Ward2017 [3], which focuses on points 2 and 3 above while enabling optimizations described in Sections 3 through 5 and delegating the removal of low-impact features to modeling efforts and keeping featurization as problem-independent as possible.

First, all 11 features relying on representation of crystal symmetry space groups with space group number floats rather than classes (e.g. using one-hot vectors) have been removed. This has been done based on the unphysical nature of such representation leading to, for instance, BCC (229) being much closer to FCC (225) than to just slightly uniaxially distorted BCC (139), which itself would be very close to trigonal structures.

Next, featurization code has been thoroughly profiled in regard to time spent on the execution of feature-specific subroutines and analyzed in the context of feature importance identified in the past work [1]. This led to the removal of the 1 *CanFormIonic* feature, which relied on combinatorically expensive guessing of oxidation states, and 3 features based on Warren–Cowley (WC) parameters [46], which were relatively very expensive without significantly contributing to the performance due to scarcity of disordered structures they aim to quantify in most atomistic datasets. However, in the future, the authors intend to include such disorder quantifying features in application-specific sets, leveraging a recently released high-performance library by Gehringer et al. [47].

Together, 15 features were removed, bringing the total number of the `KS2022` features to 256 while disproportionately improving the featurization speed. For instance, in the case of featurization of 30 sites in a disordered (no symmetry) structure, `KS2022` is 2.3 times faster than `Ward2017` (430 ms vs. 990 ms single-threaded on Apple M2 Max).

3. Optimizations for ordered structures

Modeling of disordered materials is a critical area of research [50]; however, the vast majority of atomistic ab initio datasets used for ML studies focus on highly symmetric ordered structures because of their high availability and ability to model complex phenomena in a holistic fashion if local ergodicity can be established [51,52]. One evidence of the focus on such data is the fact that out of 4.5 million atomic structures in `MPDD` [53], which includes both DFT-based [10–16,18] and experimental [21–23] data, only 54,660 or 1.25% lack any symmetry. It is also worth noting that this number used to be much lower before the recent publication of the `GNoME` dataset by Google DeepMind [18], which accounts for around $\frac{3}{4}$ of them.

In the case of remaining 98.75% structures, a 3-dimensional crystallographic spacegroup is defined for each of them along with corresponding *Wyckoff positions*, designated by letters, which are populated with either zero (empty), one (when symmetry-fixed), or up to infinitely many (typically up to a few) atoms forming a set of symmetry-equivalent sites called *crystallographic orbits* [54]. When these crystallographic orbits are collapsed into atoms occupying a unit cell, each is repeated based on the *multiplicity* associated with the Wyckoff position it occupies. These multiplicities can range from 1 up to 192 (e.g., position l in Fm-3m/225), with values 1, 2, 3, 4, 6, 8, 16, 24, 32, 48, and 96 being typical [55]. This often holds true even in compositionally simple materials like one of the experimentally observed allotropes of pure silicon with atoms at the 8a, 32e, and 96g positions [56]. For certain crystal lattice types, the multiplicity can be somewhat reduced by redefining their spatial periodicity with so-called *primitive* unit cells, like in the case of the aforementioned Si allotrope, in which primitive unit cell has 4 times fewer (34) sites but still over 10 times more than the 3 unique crystallographic orbits.

This presents an immediate and previously untapped opportunity for multiplying the computational performance of most atomistic featurizers (e.g., `Matminer` [57]) and ML models [1,3–9,58,59], which nearly always process all atoms given in the input structure occasionally converting to primitive unit cell in certain routines (`CHGNet` [7]), unless they operate on different occupancies of the same structure [60]. This allows for a proportional decrease in both CPU/GPU time and memory footprint. The general-purpose `KS2022` in `pySIPFENN` uses high-performance symmetry analysis library `spglib` [48] to automatically take advantage of this whenever possible, as depicted in the schematic in Fig. 2. It shows an interesting example of a topologically close-packed σ phase, which is critical to model in a wide range of metallic alloys [61] but challenging in terms of combinatorics because

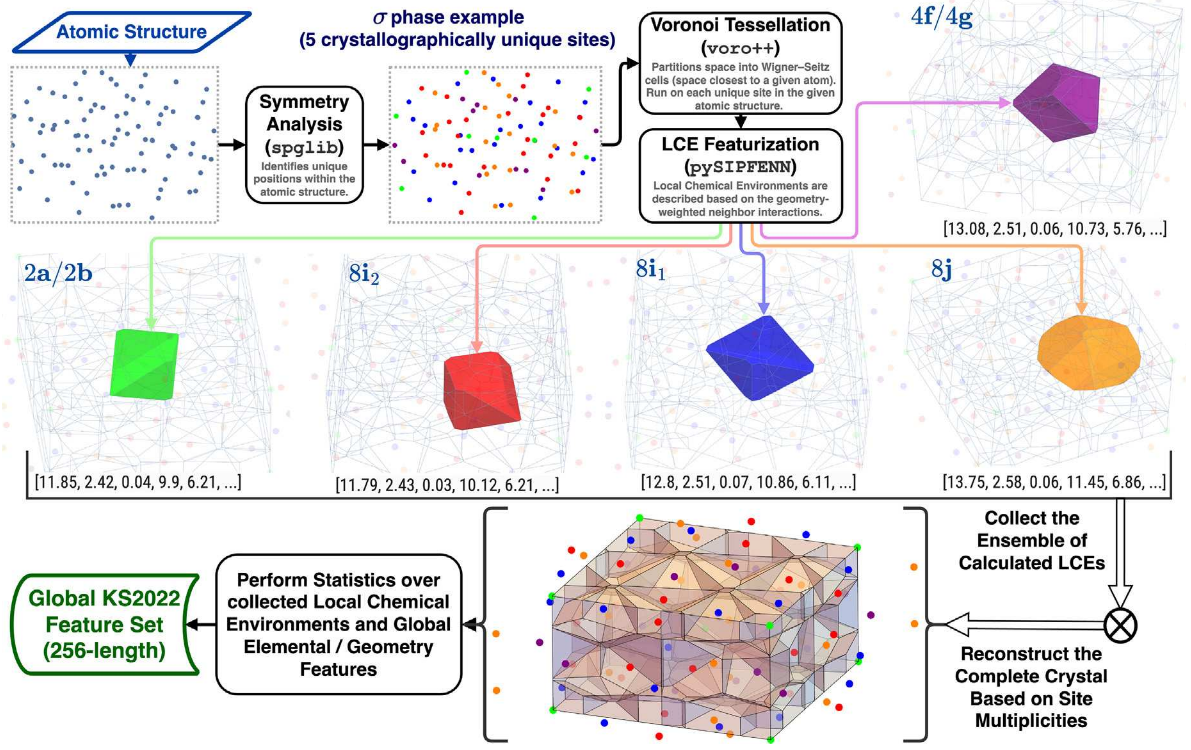

Atomic Structure
Symmetry Analysis (spglib)
Identifies unique positions within the atomic structure.
σ phase example (5 crystallographically unique sites)
Voronoi Tessellation (voro++)
Partitions space into Wigner–Seitz cells (space closest to a given atom). Run on each unique site in the given atomic structure.
LCE Featurization (pySIPFENN)
Local Chemical Environments are described based on the geometry-weighted neighbor interactions.
4f/4g
[13.08, 2.51, 0.06, 10.73, 5.76, ...]
2a/2b
8i₂
8i₁
8j
[11.85, 2.42, 0.04, 9.9, 6.21, ...]
[11.79, 2.43, 0.03, 10.12, 6.21, ...]
[12.8, 2.51, 0.07, 10.86, 6.11, ...]
[13.75, 2.58, 0.06, 11.45, 6.86, ...]
Collect the Ensemble of Calculated LCEs
Reconstruct the Complete Crystal Based on Site Multiplicities
Perform Statistics over collected Local Chemical Environments and Global Elemental / Geometry Features
Global KS2022 Feature Set (256-length)

Fig. 2. Schematic of the general-purpose `KS2022` featurization routine with built-in optimization for ordered structures. First, the atomic structure (in `pymatgen Structure` object format [32]) is loaded, and sites in it are annotated with their crystallographic orbits using `spglib` [48], denoted in the figure as unique colors. Then, one site is selected from each orbit to form a set of unique sites, for which Wigner–Seitz cells (depicted as colored polyhedra) are calculated with `Voro++` [43–45] and featurized to get site-specific local chemical environment (LCE) descriptors. The complete site ensemble is then reconstructed based on multiplicities of Wyckoff positions corresponding to the sites. A non-trivial example of σ-phase with 30 atoms belonging to 5 crystallographic orbits with interesting Wigner–Seitz cells (relative to usually shown FCC/BCC ones [49]) has been depicted.

of 5 unique sites that can be occupied by many elements [62,63] making it a very active area of ML modeling efforts [60,64] in the thermodynamics community.

In the case of KS2022 featurizer, running the same 30-atom test as in Section 2.3 but on σ phase takes on average 84 ms or is 5.1 times faster thanks to processing 6 times less sites. Similar results should be (a) quickly achievable with any other featurizer processing individual sites, including most graph representations embedding local environments (e.g., MEGNet [5]) or deconstructing graphs into graphlets (e.g., minervachem molecule featurizer [65]), and (b) possible with convolution-based models operating on graphs (e.g., ALIGNN [6]) or voxels [8] through custom adjustments to the specific convolution implementation. In the case of voxel representations and any other memory-intense ones, it may also be beneficial to utilize this approach to compress them when transferring between devices like CPU and GPU or across an HPC network.

4. Optimizations for dilute, defect, and doped structures

The optimization strategy in Section 3 ensures that only the sites that are *guaranteed* to be *crystallographically unique* are processed through featurization or graph convolution and is directly applicable to the vast majority of both data points and literature methods. However, in the case of methods relying on describing the immediate neighbors, whether through Wigner–Seitz cell (see Fig. 2) or subgraph (see, e.g., [5]), one can achieve further efficiency improvements by considering which sites are *guaranteed* to be *unique under the representation*.

There are several classes of atomic structures where the distinction above makes a difference, but the room to improve is exceptionally high when one site or small subset, in an otherwise highly symmetric structure is modified, leading to a structure that, depending on the context, will be typically called *dilute* when discussing alloys [66], *doped* when discussing electronic materials [67], or said to have *defect* in a more general sense [68]. Throughout pySIPFENN's codebase and the rest of this work, the single term *dilute* is used to refer to all of such structures due to authors' research focus on Ni-based superalloys at the time when optimizations below were open-sourced in February 2023.

To visualize the concept, one can consider, for instance, a $3 \times 3 \times 3$ body-centered cubic (BCC) conventional supercell (54 sites) and call it *base structure*. If it only contains a single specie, then KS2022 from Section 3 will recognize that there is only one crystallographic orbit and only process that one. However, if a substitution is made at any of the 54 equivalent sites, the space group will change from Im-3 m (229) to Pm-3 m (221), with 8 crystallographic orbits on 7 Wyckoff positions; thus, the default KS2022 featurizer will process 8 sites.

At the same time, several of these crystallographic orbits will be differentiated *only* by the orientation and distance to the dilute (substitution) site, which *does* affect ab initio calculation results (e.g., vacancy formation energy vs. supercell size [69]), but is *guaranteed* to *have no effect on the model's representation* because of the exact same neighborhood configuration, including angles and bond lengths, if conditions given earlier are met. Thus, it only requires adjustments to the site multiplicities or convolution implementation (simplified through, e.g., a Markov chain). In the particular dilute-BCC example at hand, depicted in Fig. 3, there are 4 such *representation-unique* crystallographic orbits. The first one, depicted in red, contains the dilute atom. The second and third, neighbor the dilute atom sharing either large hexagonal face (1st nearest neighbor shell), depicted in blue, or small square face (2nd nearest neighbor shell), depicted in orange. The fourth orbit, depicted in purple, is not affected by the dilute atom and is equivalent to the remaining 4 orbits. Thus, the total number of sites that need to be considered is reduced by a factor of 2.

The KS2022_dilute featurization routine, schematically depicted in Fig. 3, conveniently automates the above process. It does that for both simple cases, like the aforementioned substitution in pure element, and for complex cases, like introducing a dilute atom at the 2a/2b orbit in σ-phase (green cell in Fig. 2). First, the routine performs an independent identification of crystallographic orbits in the dilute structure and base structure, then identifies of the dilute site and its configuration to establish orbit equivalency under pySIPFENN's KS2022 representation, and finally reconstructs the complete ensemble of sites in the dilute structure.

In the case of KS2022_dilute implementation run on the dilute BCC supercell shown in Fig. 3, the efficiency is improved nearly proportionally to the reduction in the number of considered sites, averaging 51 ms vs. 98 ms KS2022, signifying 1.9 computational cost reduction relative to calculating all crystallographically unique sites. Or around *10-fold computational cost reduction* relative to the standard [1,3–9] approach of processing all sites (494 ms), while producing precisely the same results (within the numerical precision).

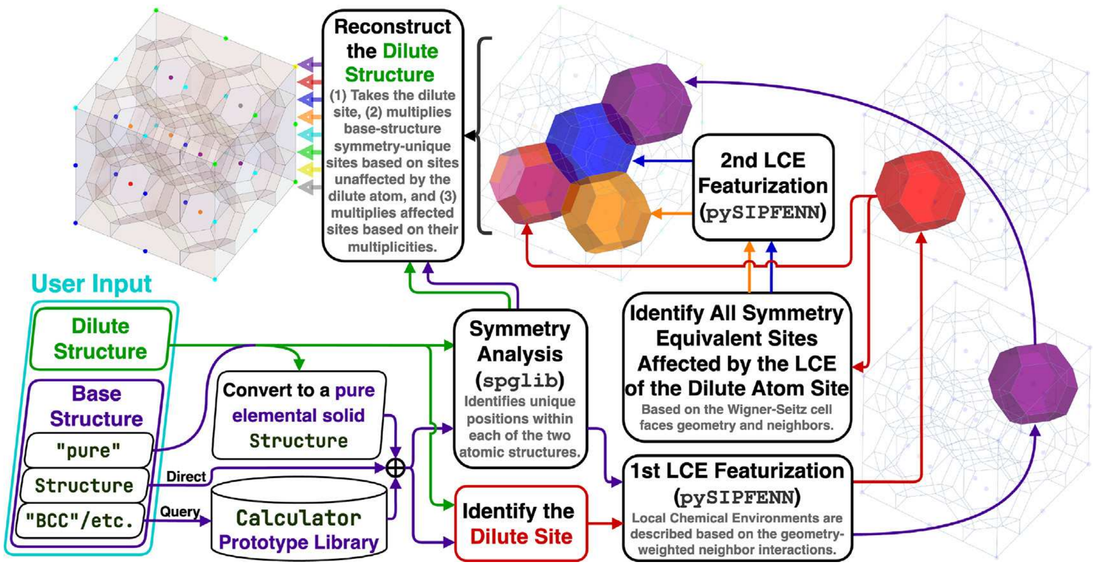

Reconstruct the Dilute Structure
(1) Takes the dilute site, (2) multiplies base-structure symmetry-unique sites based on sites unaffected by the dilute atom, and (3) multiplies affected sites based on their multiplicities.
2nd LCE Featurization (pySIPFENN)
User Input
Dilute Structure
Base Structure
"pure"
Structure
"BCC"/etc.
Direct
Query
Convert to a pure elemental solid Structure
Calculator Prototype Library
Symmetry Analysis (spglib)
Identifies unique positions within each of the two atomic structures.
Identify All Symmetry Equivalent Sites Affected by the LCE of the Dilute Atom Site
Based on the Wigner-Seitz cell faces geometry and neighbors.
Identify the Dilute Site
1st LCE Featurization (pySIPFENN)
Local Chemical Environments are described based on the geometry-weighted neighbor interactions.

Fig. 3. Core schematic of the KS2022_dilute featurizer. The dilute structure is compared to either the explicit or implicit base structure to identify the dilute site, which is then featurized alongside all crystallographically unique sites in the base structure. Information extracted from dilute structure featurization is then used to identify previously-equivalent sites affected by it, which go through the second round of featurization. Lastly, the complete ensemble is reconstructed, and KS2022 are obtained. BCC supercell is used as an example.

It is also worth mentioning the active interest of the community in this class of atomic structures, as evidenced by the dedicated ADAQ Database (defects.anyterial.se) [70] and dedicated ML modeling efforts [71] being published at the time of writing of this work.

5. Optimizations for random solid solutions

Sections 3 and 4 have demonstrated how recognition of symmetry in ordered structures can guarantee equivalency of sites and how understanding the character of featurization can further extend that notion of equivalency so that the ML representations of all sites can be obtained efficiently up to an order of magnitude faster. Random solid solutions are the conceptually opposite class of atomic structures, where the *lack of* symmetry or site equivalency is *guaranteed*, yet featurizing them requires one to solve the same problem of efficiently obtaining the ML representations of all sites present, which also happen to be infinite.

Typically, in the ab initio community, random solid solutions are represented using Special Quasirandom Structures (SQS) introduced in landmark 1990 work by Zunger et al. [72], which are *the* best structures to match neighborhood correlations in a purely random state given component ratios and number of atoms to use, hence the name *special*. For many years, finding SQS structures required exponentially complex enumeration of all choices and was limited to simple cases until another critical work by van de Walle et al. [73], which used simulated annealing Monte Carlo implemented through `ATAT` software to find these special cases much faster, exemplified through the relatively complex σ-phase and enabling the creation of SQS libraries used in thermodynamic modeling [74].

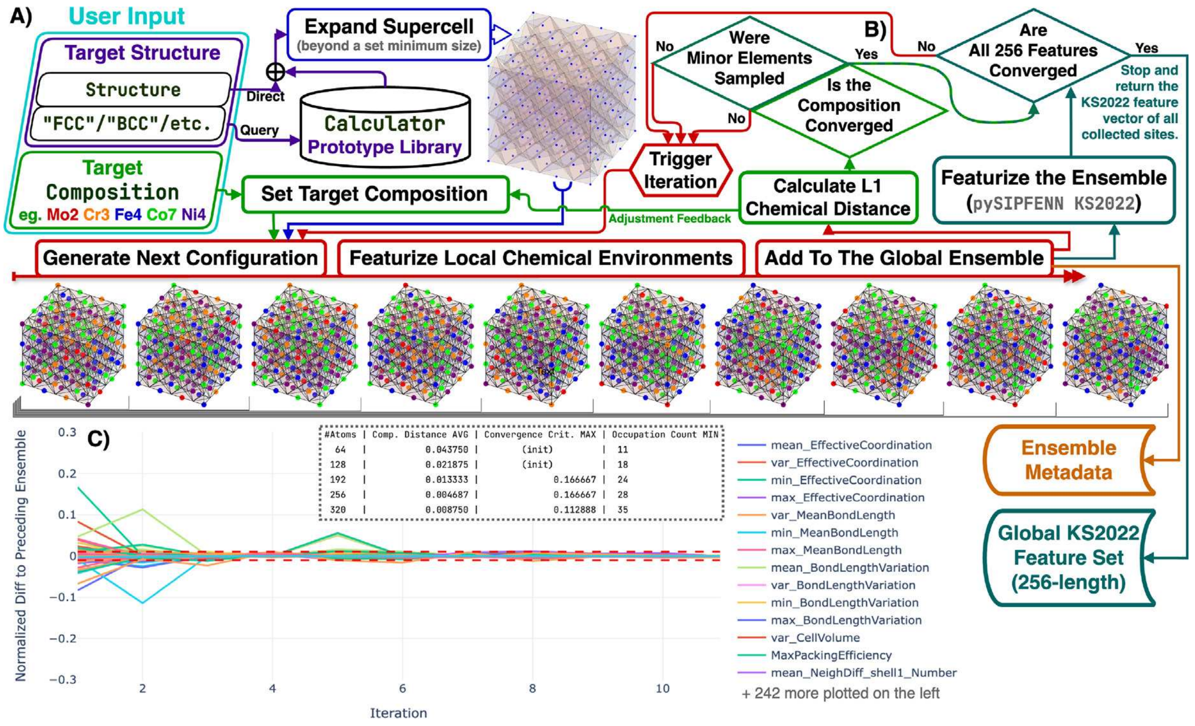
However, the direct use of an SQS may not be the optimal choice for structure-informed random solid solution featurization due to several reasons. Firstly, as discussed by van de Walle et al. [73], SQS can be expected to perform well on purely fundamental grounds for certain properties like total energy calculations, but one has to treat them with caution because different properties will depend differently on the correlation and selecting the SQS may be suboptimal. Building up on that, one could, for instance, imagine a property that depends strongly on the existence of low-frequency, high-correlation regions catalyzing a surface reaction or enabling nucleation of a dislocation. In terms of ML modeling, this notion is taken to the extreme, with calculated features being both very diverse and numerous while being expected to be universal surrogates for such mechanistically complex properties. Appendix B further explores this concept, discussing use of variability data in different applications.

Secondly, SQSs that can be generated in a reasonable time are limited in terms of the number of atoms considered, causing quantization of the composition. This is not an issue if a common grid of results is needed, e.g., to fit CALPHAD model parameters [74] or to train a single-purpose ML model [75], but it becomes a critical issue if one needs to accept an arbitrary composition as the ML model and SQS would have to be obtained every time. This issue is further amplified by the rapidly growing field of compositionally complex materials (CCMs), which exist in vast many-component compositional spaces prohibiting SQS reuse even at coarse quantizations [76] while being a popular deployment target for both forward and inverse artificial intelligence methods [77–79] due to their inherent complexity.

Based on the above, it becomes clear that costly computing of an SQS structure would have to be done for every ML model, and it would not be consistent between chemistries and complexities. At the same time, the primary motivation for limiting the number of sites for ab initio calculations is gone since `KS2022` can featurize over 1000 sites per second on a laptop (Apple M2 Max, run in parallel).

Thus, the objective of optimization is shifted towards consistency in convergence to feature vector values at infinity. To accomplish that, `pySIPFENN` goes back to random sampling but at a large scale and *individually monitoring the convergence of every feature* during the expansion procedure, implemented through `KS2022_randomSolutions` and depicted in Fig. 4, to ensure individual convergence.

Such a representation-centered approach can also efficiently account for (1) the dissimilarity of any set of chemical elements and (2) the neighbor weight during featurization, where some may be much


A)
User Input
Target Structure
Structure
"FCC"/"BCC"/etc.
Target Composition
eg. Mo2 Cr3 Fe4 Co7 Ni4
Direct
Query
Expand Supercell
(beyond a set minimum size)
Calculator
Prototype Library
Set Target Composition
B)
Were Minor Elements Sampled
No
Yes
No
Is the Composition Converged
Are All 256 Features Converged
No
Yes
Stop and return the KS2022 feature vector of all collected sites.
Trigger Iteration
Calculate L1 Chemical Distance
Adjustment Feedback
Featurize the Ensemble (pySIPFENN KS2022)
Generate Next Configuration
Featurize Local Chemical Environments
Add To The Global Ensemble
Ensemble Metadata
Global KS2022 Feature Set (256-length)
C)
Normalized Diff to Preceding Ensemble
Iteration
mean_EffectiveCoordination
var_EffectiveCoordination
min_EffectiveCoordination
max_EffectiveCoordination
var_MeanBondLength
min_MeanBondLength
max_MeanBondLength
mean_BondLengthVariation
var_BondLengthVariation
min_BondLengthVariation
max_BondLengthVariation
var_CellVolume
MaxPackingEfficiency
mean_NeighDiff_shell1_Number
+ 242 more plotted on the left

```
#Atoms | Comp. Distance AVG | Convergence Crit. MAX | Occupation Count MIN
  64   |           0.043750 |        (init)         |  11
 128   |           0.021875 |        (init)         |  18
 192   |           0.013333 |              0.166667 |  24
 256   |           0.004687 |              0.166667 |  28
 320   |           0.008750 |              0.112888 |  35
```

Fig. 4. Core schematic of the `KS2022_randomSolutions` featurizer. (A) The target structure given explicitly or implicitly is expanded to form a (lattice) (i.e. template) supercell. It is then iteratively populated with target composition (slightly adjusted each time) and divided into individual sites. (B) These are featurized, like in `KS2022`, and added to the global ensemble. The process repeats until (1) the composition is converged, (2) all species have had a chance to occur N times, and (3/C) *every individual feature* has converged to stable value over 3 consecutive iterations. Lastly, the global `KS2022` feature vector and metadata are returned. FCC supercell is used as an example in the figure.

more important than others (see highly-anisotropic σ-phase Wigner–Seitz cells in Fig. 2). It is also flexible in accepting any target structure, even a distorted one since no assumptions are made about the neighborhood geometry.

At the same time, it is important to note that such an approach is not a replacement for SQS in a general sense. It is, instead, a complementary method, as it does not result in a defined approximation of random structure but its representation for machine learning.

6. Summary and conclusions

- `pySIPFENN` or *python toolset for Structure-Informed Property and Feature Engineering with Neural Networks* is a free open-source software (FOSS) modular framework extending authors' past work [1] described in Section 1 by including many key improvements in the structure-informed featurization, machine learning model deployment, different types of transfer learning (connected to OPTIMADE API [39]), rewrite of key literature tools (e.g., `Ward2017` Java-based featurizer [3]) into Python+ `NumPy` [31], and optimizations of past feature set as described in Sections 2.1, 2.2, and 2.3.
- `pySIPFENN` framework is uniquely built from tightly integrated yet highly independent modules to allow easy use of essential functions without limiting advanced researchers from taking specific components they need, like a specific featurizer, and simply copying it into their software, reducing dependencies to the minimum (including `pySIPFENN` itself).
- Section 3 discusses how featurization of atomic structures (or configurations) to construct vector, voxel, graph, graphlet, and other representations is typically performed inefficiently because of redundant calculations and how their efficiency could be improved by considering fundamentals of crystallographic (orbits) equivalency to increase throughout of literature machine learning model, typically between 2 to 10 times (e.g., 5 times for σ-phase). Critically, this optimization applies to 98.75% of 4.5 million stored in `MPDD` [53], which includes both DFT-based [10–16,18] and experimental [21–23] data, showing massive impact if deployed. `KS2022` featurizer implements these advances in `pySIPFENN` using `spglib` [48] and `Voro++` [43–45], while retaining ability to process arbitrary structures.
- Section 4 explores how symmetry is broken in dilute, doped, and defect structures, to then discuss site equivalency under different representations and how this notion can be used to improve efficiency by skipping redundant calculations of sites which are not guaranteed to be equivalent based on crystallographic symmetry alone but need to be contrasted with defect-free representation. `KS2022_dilute` featurizer implements these advances in `pySIPFENN`. The typical speed increase users can expect is 1.5 to 3 times relative to `KS2022`. For instance, a 54-atom BCC supercell with a dilute atom, discussed in detail in Section 4, requires 2 times less site calculations relative to `KS2022` and 13.5 times less than the traditional approach of calculating all sites.
- Section 5 discusses featurization of perfectly random configuration of atoms occupying an arbitrary atomic structure and, for the first time, considers fundamental challenges with using SQS approach in the context of forward and inverse machine learning model deployment by extending past discussion on SQS limitations given by van de Walle et al. [73], which do not typically appear in ab initio and thermodynamic studies. `KS2022_randomSolutions` featurizer has been developed to efficiently featurize solid solutions of any compositional complexity by expanding the local chemical environments (LCEs) ensemble until standardized convergence criteria are met.
- As described in Section Software Availability and Accessibility, software introduced in this work is continuously tested, well documented, regularly maintained, and
- Throughout this work, the authors explicitly discuss how advances in featurization efficiency described in this work can be applied to different kinds of similar tools in the community, including those using voxel, graph, or graphlet representations.

Software availability and accessibility

`pySIPFENN` or *python toolset for Structure-Informed Property and Feature Engineering with Neural Networks* is an easily extensible free, open-source software (FOSS) under OSI-approved LGPL-3.0 license, available as (1) source code hosted in a `GitHub` repository (git.pysipfenn.org), (2) a python package through `PyPI` index, and (3) a conda package hosted through `conda-forge` channel.

It is very well-documented through (1) API reference, (2) detailed changelog, (3) install instructions, (4) tutorials and task-specific notes, and (5) FAQ, compiled for development (pysipfenn.org/en/latest), stable (pysipfenn.org/en/stable), and past (e.g., pysipfenn.org/en/v0.12.0) versions.

`pySIPFENN` has been built from the ground up to be a reliable user tool. It is automatically tested across a range of platforms (Linux/Windows / Mac (Intel) / Mac (M1)) and Python versions on every change, as well as on a weekly schedule.

It has been actively disseminated to its target audience through two large workshops organized with support from the Materials Genome Foundation (MGF / materialsgenomefoundation.org). The first one, covering `v0.10.3` and held online on March 2nd 2023, had over 300 users registered and over 100 following all exercises. It has been recorded and published on MGF's YouTube channel [80]. The second one, using `v0.12.1`, was held in-person on June 25th 2023 at the CALPHAD 2023 conference in Boston, as a part of Materials Genome Toolkit Workshops, covering its integration with `ESPEI` [25] and `pycalphad` [27]. In November 2023, it was also employed in a pair of workshop-style graduate-level guest lectures introducing materials informatics (amkrajewski.github.io/MatSE580GuestLectures), which can be used as an advanced tutorial.

CRediT authorship contribution statement

Adam M. Krajewski: Writing – original draft, Visualization, Validation, Software, Methodology, Investigation, Formal analysis, Conceptualization. **Jonathan W. Siegel:** Writing – review & editing, Validation, Supervision, Software. **Zi-Kui Liu:** Writing – review & editing, Supervision, Funding acquisition.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Zi-Kui Liu reports financial support was provided by National Science Foundation. Zi-Kui Liu reports financial support was provided by Advanced Research Projects Agency-Energy. Zi-Kui Liu reports financial support was provided by US Department of Energy Basic Energy Sciences. Zi-Kui Liu reports a relationship with Materials Genome Foundation that includes: board membership. Adam M. Krajewski reports a relationship with Materials Genome Foundation that includes: non-financial support. If there are other authors, they declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work has been funded through grants: U.S. National Science Foundation - Pathways to Enable Open-Source Ecosystems (POSE) **FAIN-2229690**, DOE Advanced Research Projects Agency-Energy (ARPA-E) **DE-AR0001435**, and U.S. Department of Energy (DOE) **DE-NE0009288**. We would like to thank (1) **Jinchao Xu** from PSU/KAUST for his contribution to the development of SIPFENN, published in [1]; (2) **Richard Otis** and **Brandon Bocklund** from **Materials Genome Foundation** for supporting this work since 2019 in a variety of ways, including invaluable guidance in organizing community workshops; (3) **Ricardo Amaral** for testing the `ModelAdjusters` submodule and reviewing the manuscript; (4) **Rushi Gong, Shuang Lin, ShunLi Shang, Hui Sun, Alexander Richter, and Luke Myers** from Phases Research Lab at PSU, **Kate Elder** from Lawrence Livermore National Lab, and others for providing feedback when testing the `pySIPFENN` software; and (5) **Jan Janssen** from Max–Planck Institute for assistance in deployment and maintenance of `pySIPFENN`'s `conda-forge` feedstock.

Appendix A. OPTIMADE property data fetching, limitations, and examples

The `OPTIMADE`-based ML-model adjuster from the `ModelAdjusters` submodule of `pySIPFENN`, introduced in Section 2.1, inherits the limitations of the current state of `OPTIMADE` [39], which include limited degree of standardization of what is served, beyond core specification (e.g., formulas and atomic structure definitions). Thus, users typically have to rely on the provider specific fields to extract data of interest, such as DFT results.

Within the current implementation of `pySIPFENN`, as explained in the corresponding documentation at pysipfenn.org, users are expected to (1) select a `provider` from the list at optimade.org, (2) consult the provider website or manual to establish which field values (scalar or array) they want to fetch for their specific use case, and (3) specify the `targetPath` to the said field. Both `provider` string and `targetPath` tuple can then be passed to `OPTIMADEAdjuster` at initialization. The provider-specific field structure can be changed at any time; thus, it should be verified each time.

Numerous fields can be queried, but to give a set of examples, at the time of writing, the following queries can be made:

- `provider='mp',targetPath=('attributes','_mp_stability','gga_gga+u','formation_energy_per_atom')` to obtain the GGA+U formation energy per atom from Materials Project [15]. If no inputs are given, this is passed as a default value.
- `provider='alexanderia',targetPath=('attributes','_alexandria_scan_formation_energy_per_atom')` for the SCAN formation energy per atom in Alexandria
- `provider='alexanderia',targetPath=('attributes','_alexandria_formation_energy_per_atom')` for the GGAsol formation energy per atom in Alexandria
- `provider='jarvis',targetPath=('attributes','_jarvis_formation_energy_peratom')` for the optb88vdw formation energy per atom in JARVIS
- `provider='mpdd',targetPath=('attributes','_mpdd_formationenergy_sipfenn_krajewski2020_novelmaterialsmodel')` for the formation energy predicted by the `SIPFENN_Krajewski2020_NovelMaterialsModel` for every structure in MPDD.

Appendix B. Random solid solution metadata and variability considerations

As mentioned in Section 5, for specific applications, one may want to track the variability in the individual additions/expansions of the global Local Chemical Environment (LCE) ensemble generated during featurization with `KS2022_randomSolutions` submodule. Information about these can be used within modeling efforts in different ways. First, they can be treated as additional meta-features passed to an ML model that may be learnt from, which can be particularly useful when modeling properties dependent on the variability in the studied material. Second, they may be used within uncertainty quantification efforts in order to estimate characteristics of the distribution, such as width or multi-modality. Third, the extreme cases or the extent of the distribution tail that may be critical to modeling properties reliant on some rare events, such as mechanical failure in the weakest point or catalytic behavior of some configurations.

Within `pySIPFENN`'s `KS2022_randomSolutions`, by default, only the final global ensemble is being returned; however, this behavior can be easily overridden with `returnMeta=True` flag passed to the main `generate_descriptor` function. As described in the documentation, this will cause it to return an extensive dictionary of fitting data as the second part of the return tuple. It can then be used to easily access data of interest in the context of variability.

First, the fitting history under `result[1]["propHistory"]` enables one to (1) track the evolution of the individual features of the growing global ensemble at different iterations, corresponding to part C of Fig. 4 in Section 5, and (2) investigate the evolution of ML predictions of one or more properties as the global ensemble is growing, as depicted in Fig. 5.

As can be seen, the formulas with sets of chemically similar elements in similar proportions, such as $Hf_6Zr_8Ta_3Nb_3$ (second from the top), generally converge quickly and do not change much during the convergence (comparable to the model's mean absolute error (MAE) of 0.050). On the contrary, complex formulas, such as $Hf_{2.5}Zr_1Ti_{25}Ta_{12.5}Nb_{26.75}V_{10}Mo_{20}Cr_2C_{0.25}$ (first from the top), take much longer to converge and can significantly change, including iteration-to-iteration, when rare components like 1/400 carbon atom in the alloy mentioned above occurs in the given ensemble addition.

Furthermore, the individual LCE ensembles generated by `KS2022_randomSolutions` can be accessed under `result[1]["individualResults"]` and investigated directly or passed through ML models. The latter is depicted in Fig. 6 showing violin plots corresponding to the property evolutions in Fig. 5.

As shown in Fig. 6, similar observations can be drawn as earlier from Fig. 5, but with improved explainability. For instance, one can observe the population of carbides in $Hf_{2.5}Zr_1Ti_{25}Ta_{12.5}Nb_{26.75}V_{10}Mo_{20}Cr_2C_{0.25}$. Furthermore, in several cases, the distribution can be observed to be relatively wide when dissimilar metals are combined and to be narrow when similar elements are combined, especially if the said similarity is gauged concerning their lattice stability.

Appendix C. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.commatsci.2024.113495.

Data availability

The source code of pySIPFENN is open under LGPL-3.0 at GitHub (git.pysipfenn.org), PyPI, and conda-forge. Documentation (pysipfenn.org), workshops, and tutorials were also published.

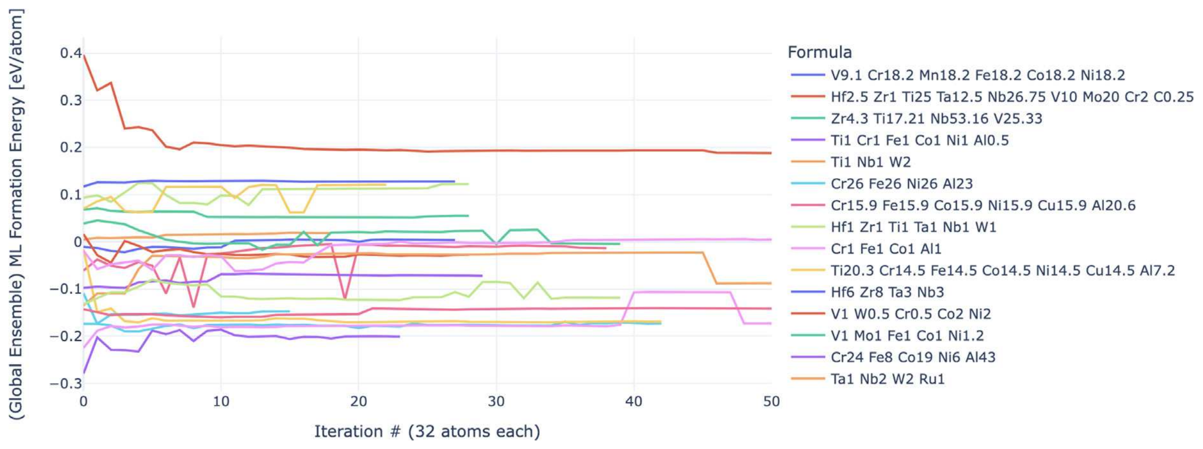


Fig. 5. An example of the evolution of ML predictions of formation energy as a function of `KS2022_randomSolutions` iteration for 20 randomly selected high entropy alloys (HEAs) from the ULTERA Database (ultera.org) [81] populating a BCC lattice.

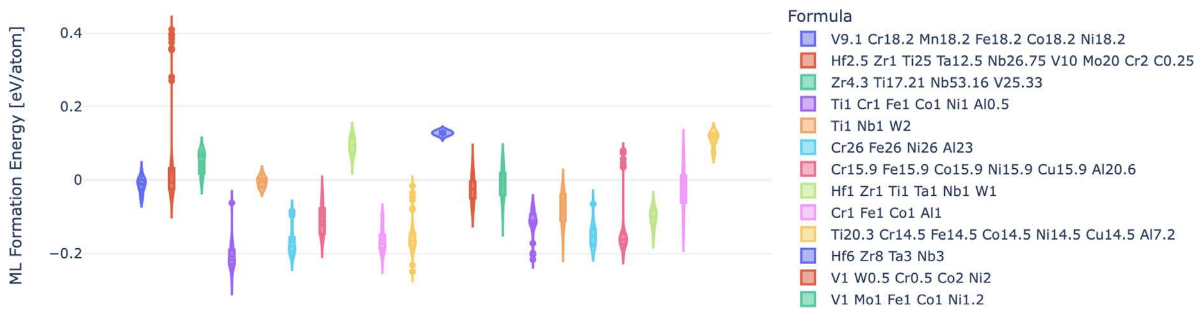


Fig. 6. An example of the fitted distributions of ML predictions of formation energy for 20 randomly selected high entropy alloys (HEAs) from the ULTERA Database (ultera.org) [81] populating a BCC lattice.

References

[1] A.M. Krajewski, J.W. Siegel, J. Xu, Z.-K. Liu, Extensible structure-informed prediction of formation energy with improved accuracy and usability employing neural networks, Comput. Mater. Sci. 208 (2022) 111254, http://dx.doi.org/10.1016/j.commatsci.2022.111254, [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0927025622000593.

[2] A. Krajewski, J. Siegel, J. Xu, Z.-K. Liu, Neural networks for structure-informed prediction of formation energy (employed in SIPFENN), 2020, http://dx.doi.org/10.5281/zenodo.4006802, [Online]. Available: https://doi.org/10.5281/zenodo.4006802.

[3] L. Ward, R. Liu, A. Krishna, et al., Including crystal structure attributes in machine learning models of formation energies via voronoi tessellations, Phys. Rev. B 96 (2) (2017) 024104, http://dx.doi.org/10.1103/PhysRevB.96.024104, [Online]. Available: http://link.aps.org/doi/10.1103/PhysRevB.96.024104.

[4] D. Jha, L. Ward, Z. Yang, et al., IRNet, in: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, ACM, New York, NY, USA, 2019, pp. 2385–2393, http://dx.doi.org/10.1145/3292500.3330703, [Online]. Available: https://dl.acm.org/doi/10.1145/3292500.3330703.

[5] C. Chen, W. Ye, Y. Zuo, C. Zheng, S.P. Ong, Graph networks as a universal machine learning framework for molecules and crystals, Chem. Mater. 31 (2019) 3572, http://dx.doi.org/10.1021/acs.chemmater.9b01294, [Online]. Available: https://pubs.acs.org/sharingguidelines.

[6] K. Choudhary, B. DeCost, Atomistic line graph neural network for improved materials property predictions, Npj Comput. Mater. 7 (1) (2021) 185, http://dx.doi.org/10.1038/s41524-021-00650-1, [Online]. Available: https://www.nature.com/articles/s41524-021-00650-1.

[7] B. Deng, P. Zhong, K. Jun, et al., CHGNet as a pretrained universal neural network potential for charge-informed atomistic modelling, Nat. Mach. Intell. 5 (2023) 1031–1041, http://dx.doi.org/10.1038/s42256-023-00716-3, [Online]. Available: https://doi.org/10.1038/s42256-023-00716-3.

[8] A. Davariashtiyani, S. Kadkhodaei, Formation energy prediction of crystalline compounds using deep convolutional network learning on voxel image representation, Commun. Mater. 4 (1) (2023) 105, http://dx.doi.org/10.1038/s43246-023-00433-9, [Online]. Available: https://www.nature.com/articles/s43246-023-00433-9.

[9] J. Schmidt, N. Hoffmann, H.-C. Wang, et al., Machine-learning-assisted determination of the global zero-temperature phase diagram of materials, Adv. Mater. 35 (22) (2023) 2210788, http://dx.doi.org/10.1002/ADMA.202210788, [Online]. Available: https://onlinelibrary.wiley.com/doi/full/10.1002/adma.202210788 https://onlinelibrary.wiley.com/doi/abs/10.1002/adma.202210788 https://onlinelibrary.wiley.com/doi/10.1002/adma.202210788.

[10] J.E. Saal, S. Kirklin, M. Aykol, B. Meredig, C. Wolverton, Materials design and discovery with high-throughput density functional theory: The open quantum materials database (OQMD), JOM 65 (11) (2013) 1501–1509, http://dx.doi.org/10.1007/s11837-013-0755-4, [Online]. Available: http://link.springer.com/10.1007/s11837-013-0755-4.

[11] S. Kirklin, J.E. Saal, B. Meredig, et al., The open quantum materials database (OQMD): assessing the accuracy of DFT formation energies, Npj Comput. Mater. 1 (1) (2015) 15010, http://dx.doi.org/10.1038/npjcompumats.2015.10, [Online]. Available: www.oqmd.org/download http://www.nature.com/articles/npjcompumats201510.

[12] J. Shen, S.D. Griesemer, A. Gopakumar, et al., Reflections on one million compounds in the open quantum materials database (OQMD), J. Phys.: Mater. 5 (3) (2022) 031001, http://dx.doi.org/10.1088/2515-7639/ac7ba9, [Online]. Available: https://iopscience.iop.org/article/10.1088/2515-7639/ac7ba9.

[13] S. Curtarolo, W. Setyawan, G.L. Hart, et al., AFLOW: An automatic framework for high-throughput materials discovery, Comput. Mater. Sci. 58 (2012) 218–226, http://dx.doi.org/10.1016/j.commatsci.2012.02.005, [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0927025612000717.

[14] C. Toher, C. Oses, D. Hicks, et al., The AFLOW fleet for materials discovery, in: Handbook of Materials Modeling, Springer International Publishing, Cham, 2018, pp. 1–28, http://dx.doi.org/10.1007/978-3-319-42913-7{_}63-1, [Online]. Available: http://link.springer.com/10.1007/978-3-319-42913-7_63-1.

[15] A. Jain, S.P. Ong, G. Hautier, et al., Commentary: The materials project: A materials genome approach to accelerating materials innovation, APL Mater. 1 (1) (2013) 011002, http://dx.doi.org/10.1063/1.4812323, [Online]. Available: http://aip.scitation.org/doi/10.1063/1.4812323.

[16] K. Choudhary, K.F. Garrity, A.C. Reid, et al., The joint automated repository for various integrated simulations (JARVIS) for data-driven materials design, Npj Comput. Mater. 6 (1) (2020) 1–13, http://dx.doi.org/10.1038/s41524-020-00440-1, [Online]. Available: https://doi.org/10.1038/s41524-020-00440-1.

[17] J. Schmidt, H.C. Wang, T.F. Cerqueira, S. Botti, M.A. Marques, A dataset of 175k stable and metastable materials calculated with the PBEsol and SCAN functionals,

Sci. Data 9 (1) (2022) 1–8, http://dx.doi.org/10.1038/s41597-022-01177-w, 2022 9:1, [Online]. Available: https://www.nature.com/articles/s41597-022-01177-w.

[18] A. Merchant, S. Batzner, S.S. Schoenholz, M. Aykol, G. Cheon, E. Dogus Cubuk, Scaling deep learning for materials discovery, 80 | Nature | 624 (2023) http://dx.doi.org/10.1038/s41586-023-06735-9, [Online]. Available: https://doi.org/10.1038/s41586-023-06735-9.

[19] S. Im, S.L. Shang, N.D. Smith, et al., Thermodynamic properties of the Nd-Bi system via emf measurements, DFT calculations, machine learning, and CALPHAD modeling, Acta Mater. 223 (2022) 117448, http://dx.doi.org/10.1016/J.ACTAMAT.2021.117448.

[20] S.-L. Shang, H. Sun, B. Pan, et al., Forming mechanism of equilibrium and non-equilibrium metallurgical phases in dissimilar aluminum/steel (Al–Fe) joints, Sci. Rep. 11 (1) (2021) 24251, http://dx.doi.org/10.1038/s41598-021-03578-0, [Online]. Available: https://www.nature.com/articles/s41598-021-03578-0.

[21] S. Gražulis, D. Chateigner, R.T. Downs, et al., Crystallography open database – an open-access collection of crystal structures, J. Appl. Crystallogr. 42 (4) (2009) 726–729, http://dx.doi.org/10.1107/S0021889809016690.

[22] S. Gražulis, A. Daškevič, A. Merkys, et al., Crystallography Open Database (COD): An open-access collection of crystal structures and platform for world-wide collaboration, Nucl. Acids Res. 40 (D1) (2012) D420–D427, http://dx.doi.org/10.1093/nar/gkr900, [Online]. Available: https://academic.oup.com/nar/article/40/D1/D420/2903497.

[23] S. Gražulis, A. Merkys, A. Vaitkus, et al., Crystallography open database: History, development, and perspectives, in: Materials Informatics, Wiley, 2019, pp. 1–39, http://dx.doi.org/10.1002/9783527802265.ch1, [Online]. Available: https://onlinelibrary.wiley.com/doi/10.1002/9783527802265.ch1.

[24] H. Sun, S.-L. Shang, R. Gong, B.J. Bocklund, A.M. Beese, Z.-K. Liu, Thermodynamic modeling of the Nb-Ni system with uncertainty quantification using PyCalphad and ESPEI, CALPHAD 82 (2023) 102563, http://dx.doi.org/10.1016/j.calphad.2023.102563, [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0364591623000354.

[25] B. Bocklund, R. Otis, A. Egorov, A. Obaied, I. Roslyakova, Z.-K. Liu, ESPEI for efficient thermodynamic database development, modification, and uncertainty quantification: application to Cu–Mg, MRS Commun. 9 (2) (2019) 618–627, http://dx.doi.org/10.1557/mrc.2019.59, [Online]. Available: https://doi.org/10.1557/mrc.2019.59 http://link.springer.com/10.1557/mrc.2019.59.

[26] G. Olson, Z. Liu, Genomic materials design: Calculation of PHAse dynamics, CALPHAD 82 (2023) 102590, http://dx.doi.org/10.1016/j.calphad.2023.102590, [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0364591623000627.

[27] R. Otis, Z.-K. Liu, Pycalphad: CALPHAD-based computational thermodynamics in python, J. Open Res. Softw. 5 (2017) 1–11, http://dx.doi.org/10.5334/jors.140.

[28] A. Zunger, Beware of plausible predictions of fantasy materials, Nature 566 (7745) (2019) 447–449, http://dx.doi.org/10.1038/d41586-019-00676-y, 2021 566:7745, [Online]. Available: https://www.nature.com/articles/d41586-019-00676-y.

[29] Q. Tao, P. Xu, M. Li, W. Lu, Machine learning for perovskite materials design and discovery, Npj Comput. Mater. 7 (1) (2021) 23, http://dx.doi.org/10.1038/s41524-021-00495-8, [Online]. Available: https://www.nature.com/articles/s41524-021-00495-8.

[30] S. Jha, M. Yen, Y.S. Salinas, E. Palmer, J. Villafuerte, H. Liang, Machine learning-assisted materials development and device management in batteries and supercapacitors: performance comparison and challenges, J. Mater. Chem. A 11 (8) (2023) 3904–3936, http://dx.doi.org/10.1039/D2TA07148G, [Online]. Available: https://xlink.rsc.org/?DOI=D2TA07148G.

[31] C.R. Harris, K.J. Millman, S.J. van der Walt, et al., Array programming with NumPy, Nature 585 (7825) (2020) 357–362, http://dx.doi.org/10.1038/s41586-020-2649-2, [Online]. Available: https://www.nature.com/articles/s41586-020-2649-2.

[32] S.P. Ong, W.D. Richards, A. Jain, et al., Python materials genomics (pymatgen): A robust, open-source python library for materials analysis, Comput. Mater. Sci. 68 (2013) 314–319, http://dx.doi.org/10.1016/j.commatsci.2012.10.028, [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0927025612006295.

[33] J. Bai, F. Lu, K. Zhang, other developers, ONNX: Open neural network exchange, 2019, [Online]. Available: https://github.com/onnx/onnx https://github.com/onnx/onnx.

[34] A. Paszke, S. Gross, F. Massa, et al., PyTorch: An imperative style, high-performance deep learning library, 2019, http://dx.doi.org/10.5555/3454287.3455008, [Online]. Available: https://dl.acm.org/doi/10.5555/3454287.3455008.

[35] I. Kalgin, A. Yanchenko, P. Ivanov, A. Goncharenko, ENOT Developers, Onnx2torch: Convert ONNX models to PyTorch, 2021, [Online]. Available: https://enot.ai/ https://github.com/ENOT-AutoDL/onnx2torch, https://enot.ai/.

[36] S. Ben-David, J. Blitzer, K. Crammer, et al., A theory of learning from different domains, Mach. Learn. 79 (2010) 151–175, http://dx.doi.org/10.1007/s10994-009-5152-4.

[37] V. Gupta, K. Choudhary, B. DeCost, et al., Structure-aware graph neural network based deep transfer learning framework for enhanced predictive analytics on diverse materials datasets, Npj Comput. Mater. 10 (1) (2024) 1, http://dx.doi.org/10.1038/s41524-023-01185-3, [Online]. Available: https://www.nature.com/articles/s41524-023-01185-3.

[38] C.W. Andersen, R. Armiento, E. Blokhin, et al., OPTIMADE, an API for exchanging materials data, Sci. Data 8 (1) (2021) 217, http://dx.doi.org/10.1038/s41597-021-00974-z, [Online]. Available: https://www.nature.com/articles/s41597-021-00974-z.

[39] M. Evans, J. Bergsma, A. Merkys, et al., Developments and applications of the OPTIMADE API for materials discovery, design, and data exchange, Digit. Discov. (2024) http://dx.doi.org/10.1039/D4DD00039K, [Online]. Available: http://pubs.rsc.org/en/Content/ArticleLanding/2024/DD/D4DD00039K.

[40] M.L. Evans, C.W. Andersen, S. Dwaraknath, M. Scheidgen, A. Fekete, D. Winston, Optimade-python-tools: a Python library for serving and consuming materials data via OPTIMADE APIs, J. Open Sour. Softw. 6 (65) (2021) 3458, http://dx.doi.org/10.21105/JOSS.03458, [Online]. Available: https://joss.theoj.org/papers/10.21105/joss.03458.

[41] L. Torrey, J. Shavlik, in: E.S. Olivas, J.D.M. Guerrero, M. Martinez-Sober, J.R. Magdalena-Benedito, A.J. Serrano López (Eds.), Handbook of Research on Machine Learning Applications and Trends - Chapter 11: Transfer Learning, IGI Global, 2010, http://dx.doi.org/10.4018/978-1-60566-766-9, [Online]. Available: http://services.igi-global.com/resolvedoi/resolve.aspx?doi=10.4018/978-1-60566-766-9.

[42] F. Pedregosa Fabianpedregosa, V. Michel, O. Grisel Oliviergrisel, et al., Scikit-learn: Machine learning in python, J. Mach. Learn. Res. 12 (85) (2011) 2825–2830, [Online]. Available: http://jmlr.org/papers/v12/pedregosa11a.html.

[43] C.H. Rycroft, Multiscale modeling in granular flow, 2007, [Online]. Available: https://dspace.mit.edu/handle/1721.1/41557.

[44] C. Rycroft, Voro++: a Three-Dimensional Voronoi Cell Library in C++, Tech. Rep., Lawrence Berkeley National Laboratory (LBNL), Berkeley, CA, 2009, http://dx.doi.org/10.2172/946741, [Online]. Available: http://www.osti.gov/servlets/purl/946741-A8FxbI/.

[45] J. Lu, E.A. Lazar, C.H. Rycroft, An extension to Voro++ for multithreaded computation of Voronoi cells, Comput. Phys. Comm. 291 (2023) 108832, http://dx.doi.org/10.1016/j.cpc.2023.108832, [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0010465523001777.

[46] J.M. Cowley, An approximate theory of order in alloys, Phys. Rev. 77 (5) (1950) 669, http://dx.doi.org/10.1103/PhysRev.77.669, [Online]. Available: https://journals.aps.org/pr/abstract/10.1103/PhysRev.77.669.

[47] D. Gehringer, M. Friák, D. Holec, Models of configurationally-complex alloys made simple, Comput. Phys. Comm. 286 (27) (2023) 108664, http://dx.doi.org/10.1016/j.cpc.2023.108664, [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0010465523000097.

[48] A. Togo, I. Tanaka, Spglib: a software library for crystal symmetry search, 2018, [Online]. Available: http://arxiv.org/abs/1808.01590.

[49] J. Bohm, M. Bohm, R.B. Heimann, Voronoi polyhedra: A useful tool to determine the symmetry and bravais class of crystal lattices, Cryst. Res. Technol. 31 (8) (1996) 1069–1075, http://dx.doi.org/10.1002/CRAT.2170310816.

[50] M. Zaki, A. Jan, N.M.A. Krishnan, J.C. Mauro, Glassomics: An omics approach toward understanding glasses through modeling, simulations, and artificial intelligence, MRS Bull. 48 (10) (2023) 1026–1039, http://dx.doi.org/10.1557/s43577-023-00560-1, [Online]. Available: https://link.springer.com/10.1557/s43577-023-00560-1.

[51] Z.-K. Liu, Theory of cross phenomena and their coefficients beyond Onsager theorem, Mater. Res. Lett. 10 (7) (2022) 393–439, http://dx.doi.org/10.1080/21663831.2022.2054668, [Online]. Available: https://www.tandfonline.com/doi/full/10.1080/21663831.2022.2054668.

[52] Z.-K. Liu, Thermodynamics and its prediction and CALPHAD modeling: Review, state of the art, and perspectives, CALPHAD 82 (2023) 102580, http://dx.doi.org/10.1016/j.calphad.2023.102580, [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0364591623000524.

[53] A.M. Krajewski, J.W. Siegel, S.-L. Shang, Y. Wang, J. Xu, Z.-K. Liu, MPDD: The material-property-descriptor database, 2021, [Online]. Available: https://phaseslab.com/mpdd.

[54] U. Müller, Remarks on Wyckoff positions, in: International Tables for Crystallography, International Union of Crystallography, Chester, England, 2006, pp. 24–26, http://dx.doi.org/10.1107/97809553602060000539, [Online]. Available: https://onlinelibrary.wiley.com/iucr/itc/A1a/ch1o3v0001/.

[55] M.J. Mehl, D. Hicks, C. Toher, et al., The AFLOW library of crystallographic prototypes: Part 1, Comput. Mater. Sci. 136 (2017) S1–S828, http://dx.doi.org/10.1016/j.commatsci.2017.01.017, [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0927025617300241.

[56] J. Gryko, P.F. McMillan, R.F. Marzke, et al., Low-density framework form of crystalline silicon with a wide optical band gap, Phys. Rev. B 62 (12) (2000) R7707–R7710, http://dx.doi.org/10.1103/PhysRevB.62.R7707, [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevB.62.R7707.

[57] L. Ward, A. Dunn, A. Faghaninia, et al., Matminer: An open source toolkit for materials data mining, Comput. Mater. Sci. 152 (2018) 60–69, http://dx.doi.org/10.1016/j.commatsci.2018.05.018, [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0927025618303252.

[58] S. Banik, K. Balasubramanian, S. Manna, S. Derrible, S.K. Sankaranarayananan, Evaluating generalized feature importance via performance assessment of machine learning models for predicting elastic properties of materials, Comput. Mater. Sci. 236 (2024) 112847, http://dx.doi.org/10.1016/j.commatsci.2024.112847, [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0927025624000685.

[59] M. Hu, J. Yuan, T. Sun, M. Huang, Q. Liang, Atomtransmachine: An atomic feature representation model for machine learning, Comput. Mater. Sci. 200 (2021) 110841, http://dx.doi.org/10.1016/J.COMMATSCI.2021.110841.

[60] J.-C. Crivello, J.-M. Joubert, N. Sokolovska, Supervised deep learning prediction of the formation enthalpy of complex phases using a DFT database: The σ-phase as an example, Comput. Mater. Sci. 201 (2022) 110864, http://dx.doi.org/10.1016/j.commatsci.2021.110864, [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0927025621005796.

[61] J.-M. Joubert, Crystal chemistry and Calphad modeling of the σ phase, Prog. Mater. Sci. 53 (3) (2008) 528–583, http://dx.doi.org/10.1016/j.pmatsci.2007.04.001, [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0079642507000242.

[62] W.-M. Choi, Y.H. Jo, D.G. Kim, S.S. Sohn, S. Lee, B.-J. Lee, A thermodynamic description of the Co-Cr-Fe-Ni-V system for high-entropy alloy design, CALPHAD 66 (2019) 101624, http://dx.doi.org/10.1016/j.calphad.2019.05.001, [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0364591618302165.

[63] M. Ostrowska, G. Cacciamani, Thermodynamic modelling of the σ and μ phases in several ternary systems containing Co, Cr, Fe, Mo, Re and W, J. Alloys Compd. 845 (2020) 156122, http://dx.doi.org/10.1016/J.JALLCOM.2020.156122.

[64] Y. Zha, W. Liu, J. Fan, L. Jiang, Y. Li, X.G. Lu, Applying enhanced active learning to predict formation energy, Comput. Mater. Sci. 235 (2024) 112825, http://dx.doi.org/10.1016/J.COMMATSCI.2024.112825.

[65] M. Tynes, M.G. Taylor, J. Janssen, et al., Linear graphlet models for accurate and interpretable cheminformatics, 2024, http://dx.doi.org/10.26434/chemrxiv-2024-r81c8, [Online]. Available: https://chemrxiv.org/engage/chemrxiv/article-details/65d9282fe9ebbb4db916761e.

[66] X. Chong, S.L. Shang, A.M. Krajewski, et al., Correlation analysis of materials properties by machine learning: illustrated with stacking fault energy from first-principles calculations in dilute fcc-based alloys, J. Phys.: Condens. Matter. 33 (29) (2021) 295702, http://dx.doi.org/10.1088/1361-648X/AC0195, [Online]. Available: https://iopscience.iop.org/article/10.1088/1361-648X/ac0195.

[67] T. Chen, Y. Yuan, X. Mi, et al., Interaction of elements in dilute Mg alloys: a DFT and machine learning study, J. Mater. Res. Technol. 21 (2022) 4512–4525, http://dx.doi.org/10.1016/J.JMRT.2022.11.071.

[68] C.W.M. Castleton, A. Höglund, S. Mirbt, Density functional theory calculations of defect energies using supercells, Modelling Simul. Mater. Sci. Eng. 17 (8) (2009) 084003, http://dx.doi.org/10.1088/0965-0393/17/8/084003, [Online]. Available: https://iopscience.iop.org/article/10.1088/0965-0393/17/8/084003.

[69] C.Z. Hargather, J.M. O'Connell, A systematic first-principles study of computational parameters affecting self-diffusion coefficients in FCC Ag, Cu, and Ni, J. Phase Equilib. Diffusion (2022) http://dx.doi.org/10.1007/s11669-022-00991-4, [Online]. Available: https://link.springer.com/10.1007/s11669-022-00991-4.

[70] J. Davidsson, V. Ivády, R. Armiento, I.A. Abrikosov, ADAQ: Automatic workflows for magneto-optical properties of point defects in semiconductors, Comput. Phys. Comm. 269 (2021) 108091, http://dx.doi.org/10.1016/j.cpc.2021.108091, [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0010465521002034.

[71] M.H. Rahman, P. Gollapalli, P. Manganaris, et al., Accelerating defect predictions in semiconductors using graph neural networks, APL Mach. Learn. 2 (1) (2024) 16122, http://dx.doi.org/10.1063/5.0176333/3279661, [Online]. Available: /aip/aml/article/2/1/016122/3279661/Accelerating-defect-predictions-in-semiconductors.

[72] A. Zunger, S.-H. Wei, L.G. Ferreira, J.E. Bernard, Special quasirandom structures, Phys. Rev. Lett. 65 (3) (1990) 353–356, http://dx.doi.org/10.1103/PhysRevLett.65.353, [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevLett.65.353.

[73] A. van de Walle, P. Tiwary, M. de Jong, et al., Efficient stochastic generation of special quasirandom structures, CALPHAD 42 (2013) 13–18, http://dx.doi.org/10.1016/j.calphad.2013.06.006, [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0364591613000540.

[74] A. van de Walle, R. Sun, Q.J. Hong, S. Kadkhodaei, Software tools for high-throughput CALPHAD from first-principles data, CALPHAD 58 (2017) 70–81, http://dx.doi.org/10.1016/J.CALPHAD.2017.05.005.

[75] C. Tandoc, Y.-J. Hu, L. Qi, P.K. Liaw, Mining of lattice distortion, strength, and intrinsic ductility of refractory high entropy alloys, Npj Comput. Mater. 9 (1) (2023) 53, http://dx.doi.org/10.1038/s41524-023-00993-x, [Online]. Available: https://www.nature.com/articles/s41524-023-00993-x.

[76] A.M. Krajewski, A.M. Beese, W.F. Reinhart, Z.-K. Liu, Efficient generation of grids and traversal graphs in compositional spaces towards exploration and path planning exemplified in materials, 2024, http://dx.doi.org/10.48550/arXiv.2402.03528, [Online]. Available: http://arxiv.org/abs/2402.03528.

[77] A.A. Catal, E. Bedir, R. Yilmaz, et al., Machine learning assisted design of novel refractory high entropy alloys with enhanced mechanical properties, 2023, http://dx.doi.org/10.1016/j.commatsci.2023.112612, [Online]. Available: https://doi.org/10.1016/j.commatsci.2023.112612.

[78] Z. Rao, P.-Y. Tung, R. Xie, et al., Machine learning–enabled high-entropy alloy discovery, Science 378 (6615) (2022) 78–85, http://dx.doi.org/10.1126/science.abo4940, [Online]. Available: https://www.science.org/doi/10.1126/science.abo4940.

[79] A. Debnath, L. Raman, W. Li, et al., Comparing forward and inverse design paradigms: A case study on refractory high-entropy alloys, J. Mater. Res. 38 (17) (2023) 4107–4117, http://dx.doi.org/10.1557/s43578-023-01122-6, [Online]. Available: https://link.springer.com/10.1557/s43578-023-01122-6.

[80] A. Krajewski, 2023 03 02 MGF workshop on pySIPFENN by Adam Krajewski - YouTube, 2023, [Online]. Available: https://www.youtube.com/watch?v=OHgkRuE0UQM.

[81] A. Debnath, A.M. Krajewski, H. Sun, et al., Generative deep learning as a tool for inverse design of high entropy refractory alloys, J. Mater. Inform. 1 (1) (2021) 3, http://dx.doi.org/10.20517/jmi.2021.05, [Online]. Available: https://www.oaepublish.com/articles/jmi.2021.05.