Robust recognition and exploratory analysis of crystal structures using machine learning

Dissertation

zur Erlangung des akademischen Grades

Doctor rerum naturalium
(Dr. rer. nat.)

im Fach: Physik

Spezialisierung: Theoretische Physik

eingereicht an der Mathematisch-Naturwissenschaftlichen Fakultät
der Humboldt-Universität zu Berlin

von

M.Sc. Andreas Leitherer

Gutachter/innen:	1. Prof. Dr. Matthias Scheffler
	2. Prof. Dr. Dr. h.c. Claudia Draxl
	3. Dr. Sergei V. Kalinin

Tag der mündlichen Prüfung: 25.05.2022

Abstract

Artificial intelligence (AI) has the potential to spark significant advances in multiple scientific fields. In materials science, AI tools are driving a paradigm shift towards big data-centric research. Large computational databases with millions of entries and high-resolution experiments such as electron microscopy contain large (and growing) amount of information. To leverage this under-utilized – yet very valuable – data towards discovering hidden patterns and eventually novel physics, automatic analytical methods need to be developed. Artificial neural networks are attractive candidates due to their ability to recognize complex patterns. The classification of the crystal structure of a material is essential for its characterization. Given an atomic structure, i.e., atomic coordinates labeled by chemical-species symbols, the goal is to assign the most similar crystal structure (the so-called structural class) that is currently known. The available data is structurally diverse but often defective and incomplete. Specifically, theoretical or experimental limitations result in large atomic displacements or even substantial amount of missing atoms. A suitable method should therefore be robust with respect to sources of inaccuracy, while being able to treat multiple systems. Available methods are either robust or can treat a large number of systems, but do not fulfill both criteria at the same time. Moreover, human intervention and case-by-case hand-tuning of arbitrary tolerance parameters is often required to arrive at a satisfactory classification, which is undesirable for big-data applications. In this work, we introduce *ARISE*, a Bayesian-deep-learning based framework that can treat more than 100 structural classes in robust fashion, without any predefined threshold. The selection of structural classes, which can be easily extended on demand, encompasses a wide range of materials, in particular, not only bulk but also two- and one-dimensional systems. For the local study of large and possibly polycrystalline samples, we extend ARISE by introducing *strided pattern matching (SPM)*. While being trained on ideal structures only, ARISE correctly characterizes strongly perturbed single- and polycrystalline systems, from both synthetic and experimental resources. The presented applications traverse multiple dimensions, including one- (carbon nanotubes), two- (graphene and many other examples), three- (synthetic bulk systems and nanoparticle measurements), and four-dimensional studies (time-resolved measurement of nanoparticles). The probabilistic nature of the Bayesian-deep-learning model allows to obtain principled uncertainty estimates which are found to be correlated with crystalline order of metallic nanoparticles in electron-tomography experiments. Applying unsupervised learning to the internal neural-network representations reveals grain boundaries and (unapparent) structural regions sharing easily interpretable geometrical properties. The unsupervised analysis allows to *explain* the trained model, addressing the topic of *explainability* in machine learning. This work enables the hitherto hindered analysis of noisy atomic structural data.

Zusammenfassung

Künstliche Intelligenz besitzt großes Potential eine Vielfalt an wissenschaftlichen Feldern signifikant zu beeinflussen. In der Materialwissenschaft läuten Methoden der Künstlichen-Intelligenz wie z.B. künstliche neuronale Netze, einen Paradigmenwechsel in Richtung Big-data zentrierter Forschung ein. Datenbanken enormer Größe mit Millionen von Einträgen, sowie hochauflösende Experimente, z.B. Elektronenmikroskopie, enthalten ein Fülle an wachsender Information. Um diese ungenützten und doch wertvollen Daten für die Entdeckung verborgener Muster und letztendlich neuartiger Physik zu benutzen, ist die Entwicklung automatischer analytischer Methoden notwendig. Die Klassifizierung der Kristallstruktur eines Materials ist essentiell für dessen Charakterisierung. Gegeben eine atomare Struktur, d.h. eine Liste von Koordinaten und deren dazugehörige chemische Identitäten, ist das Ziel die ähnlichste momentan bekannte Kristallstruktur, die sogenannte Strukturklasse, zuzuweisen. Die vorhandenen Kristallstruktur-Daten sind reichhaltig in der Komplexität der räumlichen Anordnung der Atome, aber enthalten oft Defekte und sind unvollständig definiert. Speziell sorgen theoretische oder experimentelle Einschränkungen dafür, dass Atome räumlich stark versetzt sind oder in großen Teilen fehlen. Eine geeignete Methode sollte daher robust bezüglich dieser Ungenauigkeiten sein, allerdings gleichzeitig die Fähigkeit besitzen viele Systeme zu klassifizieren. Die verfügbaren Methoden sind entweder robust, oder können eine große Anzahl an Systemen behandeln - erfüllen jedoch nicht beide Kriterien gleichzeitig. Weiterhin ist es bei einigen Methoden nötig, beliebig wählbare Toleranz-Parameter per Hand und von Fall zu Fall anzupassen, um eine zufriedenstellende Klassifizierung zu erreichen. Dies ist vor allem von Nachteil bei Anwendungen auf große Datensätze. In dieser Arbeit führen wir *ARISE* ein, eine Methode basierend auf Bayesian deep learning, die mehr als 100 Strukturklassen robust und ohne festzulegende Schwellwerte klassifizieren kann. Die getroffene Auswahl an Strukturklassen, die einfach erweitert werden kann, enthält ein breites Spektrum an Materialien. Insbesondere sind nicht nur Bulk-Materialien enthalten, sondern auch zwei- und ein-dimensionale Systeme. Für die lokale Untersuchung von großen und möglicherweise polykristallinen Systemen, erweitern wir ARISE über die "strided pattern matching" Methode. Obwohl nur auf perfekte Strukturen trainiert, kann ARISE stark gestörte einfach- und polykristalline Systeme synthetischen als auch experimentellen Ursprungs charakterisieren. Die diskutierten Anwendungen umfassen mehrere Dimensionen, insbesondere ein-dimensionale (Kohlenstoffnanoröhren), zwei-dimensionale (Graphen sowie eine Vielzahl anderer Beispiele), drei-dimensionale (synthetische Bulk Systeme und Messungen von Nanopartikeln), sowie vier-dimensionale Analysen (zeitaufgelöste Messungen von Nanopartikeln). Das Model basiert auf Bayesian deep learning und ist dadurch probabilistisch, was die Berechnung von Unsicherheiten auf systematische Art und Weise erlaubt. Wir zeigen insbesondere, dass diese Unsicherheiten mit der Kristallordnung von metallischen Nanopartikeln in Elektronentomographie-Experimenten korrelieren. Die Anwendung von unüberwachtem Lernen auf die internen Darstellungen des neuronalen Netzes enthüllt Korngrenzen und nicht ersichtliche Struktur-Regionen, die über einfach interpretierbare geometrische Eigenschaften miteinander verknüpft sind. Diese unüberwachte Analyse ermöglicht das trainierte Modell zu *erklären* und geht insbesondere das Themenfeld *explainability* innerhalb des maschinellen Lernens an. Diese Arbeit ermöglicht die Analyse atomarer Strukturen mit starken Rauschquellen auf eine bisher nicht mögliche Weise.

Acknowledgements

First, I want to thank Prof. Dr. Matthias Scheffler for enabling and conceptualizing the project and research direction. In particular, I am grateful for having the chance to work at the Fritz Haber Institute and to make use of the fantastic resources – both human and computational – that came along with this experience. Moreover, I want to thank very much Prof. Dr. Claudia Draxl for agreeing to be my supervisor and supporting my thesis, allowing me to finish my PhD at the Humboldt Universität zu Berlin.

A major thank goes to Dr. Luca Ghiringhelli for the supervision which included countless meetings and discussions which have been inspiring and of direct use for the project and in general for my scientific and general future. Extensive thanks goes to Dr. Angelo Ziletti, which together with Luca has been the closest contact for me from day one, introducing me to machine learning and in general providing me with countless useful tips for immediate and long term problems. These remarks apply to both Luca and Angelo, and I am extremely grateful to being given this opportunity, experiencing strong support and advice throughout my PhD, which I assure is not taken for granted and recognized and valued highly by myself. Notably, I feel like I have been granted a (scientific) revelation, lifting me from the status of a student and bringing me closer to research life. I do not judge myself but hope that with this work I am also objectively much closer to being a researcher. I have been told that (the following is a citation, the original author may be guessed) instead of keep thinking in terms of simple, catchy explanations, one should rather embrace the complexity and try to ride the wave. While some want to "solve" a puzzle rather than provide tools for puzzles one may have not yet dreamed of.

I want to thank all people that I met at FHI and Humboldt, accepting me in their communities and having fun experiences and exchanges during coffee breaks, BiGmax conferences etc. etc. – this list of people in particular includes Emre Ahmetcik, Lucas Foppa, Benjamin Regler, Marcel Langer, Florian Knoop, Yair Litman, Marcin Kryński, Maja-Olivia Lenz, Luigi Sbailò at FHI as well as Martin Kuban, Maria Troppenz, Benedikt Hoock, Daniel Speckhard at HU plus many more people, also from other institutes, that I had the chance to meet and discuss with during my time at FHI. In particular, I want to thank Dr. Christian Liebscher from the Max Planck Institute for iron research in Düsseldorf, notably for providing important, vital insight and discussion towards the end of my PhD.

People who supported me prior to my PhD shall not be left unnamed, in particular, Prof. Kristina Giesel and Prof. Hanno Sahlmann (FAU Erlangen Nuremberg), and Prof. Gerd Leuchs (Max Planck institute for the science of light at Erlangen) who supported me during my search for a PhD and have been supervisors during my Master and Bachelor projects. Further close collaborators that I want to mention and explicitly thank are for my Bachelor time Dr. Christoph Marquardt, Gerhard Schunk, Michael Förtsch, Imran Khan, and for my Master David Winnekens and all other members of the quantum-gravity department.

Die größte aller erdenklichen Danksagungen geht jedoch an meine Familie. Dies schließt meine geliebten Eltern ein sowie meine geliebten Geschwister Peter, Susi, und Biene als auch unseren Familienzuwachs Nico und Sven. Zu guter letzt bedanke ich mich bei allen die mich auf meinem Weg bisher unterstützt haben, ich hoffe ihr erkennt unter anderem anhand dieser Dissertation die Früchte eurer Arbeit.

Contents

List of Abbreviations and Symbols

Abbreviations

- ARISE = Artificial-Intelligence-based Structure Evaluation
- SPM = Strided Pattern Matching
- AI = Artificial Intelligence
- AGI = Artificial General Intelligence
- ML = Machine Learning
- DL = Deep Learning
- ANN(s) = Artificial Neural Network(s)
- NN(s) = Neural Network(s)
- MLP(s) = Multilayer Perceptron(s)
- ReLU = Rectified Linear Unit
- SGD = Stochastic Gradient Descent
- GMM(s) = Gaussian Mixture Model(s)
- TPE = Tree-structured Parzen estimator
- KL = Kullback–Leibler
- MC = Monte Carlo
- 1D = One-Dimensional
- 2D = Two-Dimensional
- 3D = Three-Dimensional
- 4D = Four-Dimensional
- fcc = face-centered cubic

- bcc = body-centered cubic
- hcp = hexagonal close-packed
- dhcp = double hexagonal close-packed
- CNT(s) = Carbon Nanotube(s)
- SOAP = Smooth Overlap of Atomic Positions
- HDBSCAN = Hierarchical Density-based Spatial Clustering Applications with Noise
- PCA = Principal Component Analysis
- t-SNE = t-distributed Stochastic Neighborhood Embedding
- UMAP = Uniform Manifold Approximation and Projection
- PTM = Polyhedral Template Matching
- CNA = Common Neighbor Analysis
- a-CNA = Adaptive Common Neighbor Analysis
- BAA = Bond Angle Analysis
- APT = Atom-Probe Tomography
- STEM = Scanning Transmission Electron Microscopy
- HAADF = High-Angle Annular Dark-Field Imaging
- HRTEM = High-Resolution Transmission Electron Microscopy
- AET = Atomic Electron Tomography
- i.i.d. = Independent and Identically Distributed

Symbols

- a Scalar
- **a** Vector
- **A** Matrix
- **W** Weight matrix
- D Dataset
- **X** Dataset inputs

- $\mathbf{Y}$ Dataset targets
- $\mathbf{x}^{(i)}$ Input training example i
- $\mathbf{y}^{(i)}$ Target training example i
- $\mathbf{y}_j^{(i)}$ Class j for target training example i
- $\boldsymbol{\omega}$ Model parameters

Chapter 1

Introduction

Since its establishment as a scientific field in the 1950s, Artificial Intelligence (AI) has gone through several waves of enthusiasm and disappointment [1, 2]. The recent surge in interest is due to a variety of advancements both in recent time and the last couple of centuries. One major reason is the advancement of computational resources, i.e., new levels of parallelism reached in high-performance computing [3], the introduction of graphics [4] and tensor [5] processing units, and – in the near future – exascale computing. Moreover, big datasets have become available in the course of digitalization (ImageNet [6], MNIST [7]) and multiple algorithmic innovations have been made [8–11]. Especially in deep learning (DL) [12–14] these advancements have enabled several breakthroughs, including record-breaking performance in image [15] and speech [16] recognition, machine translation [17], the mastering of highly complex games such as Go [18, 19], and the generation of photo-realistic images [11]. These developments have strongly influenced the physical sciences [20] including materials science [21]. In particular, big data-centric materials science is arising as a new paradigm in materials research [22–25]. Several applications of AI in materials science are being explored, e.g., for materials property prediction. A general goal is to discover hidden patterns and trends in data, ultimately yielding new physical insights that enable the design of novel materials with application-tailored properties. The main topic of this thesis – characterizing the crystal structure of a given material – is a crucial step in materials characterization. In particular, deep neural networks have already been demonstrated to be powerful in this setting [26].

In this introductory chapter, we first provide an overview of AI and its subfields in section 1.1, where we stress the main ideas behind the individual subdivisions and the AI techniques used in this work. Section 1.2 provides a short overview of deep-learning applications in materials science, a rapidly developing field. Then, we specialize to the problem of crystal-structure recognition in section 1.3, where we introduce and motivate the general concept of crystal-structure classification and how previous work can be improved using AI and in particular deep learning. Finally, section 1.4 discusses the importance of so-called descriptors in materials science and specifically crystal-structure identification.

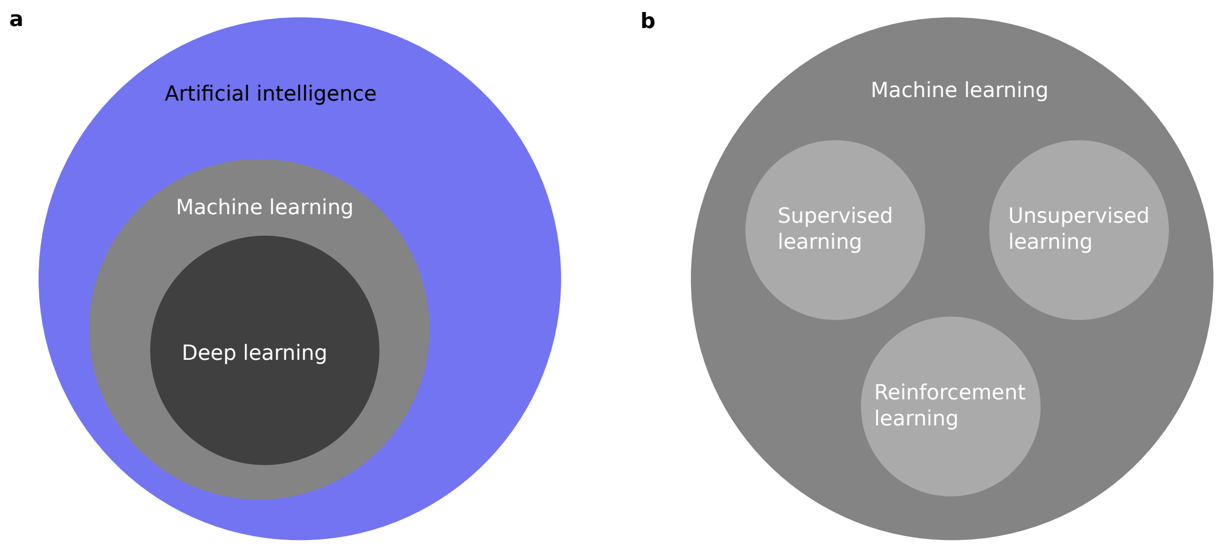

Figure 1.1: Sketch of two common subdivisions of AI (**a**) and machine learning (**b**).

1.1 Artificial intelligence and its subfields

There are multiple ways to classify and subdivide AI. On the highest level, one may distinguish between "narrow AI" and Artificial General Intelligence (AGI) [1, 27]. AGI is aiming at the development of AI systems that operate fully autonomously and solve various complex problems that would normally require human intelligence (image and speech recognition, autonomous driving) or even exceed it. Ideally, AGI systems should generalize across different environments that are not part of their original training. Note that defining and quantifying "intelligence" in machines is an involved topic that has intersections with various fields such as philosophy and psychology [27–30]. In this thesis and generally in contemporary AI research, "narrow AI" is the main focus, i.e., the development of tools that are specialized to certain tasks and usually do not generalize across different domains (e.g., an AI system that knows how to play chess will not know how to autonomously drive a car – without adding new information and/or retraining). While such AI systems may sound too specialized, the benefits become apparent from their record-breaking performance on tasks that are difficult to formalize in a mathematical description – even though these tasks may seem intuitive for humans (e.g., face recognition in images) [14].

A common subdivision of AI is sketched in Fig. 1.1a. One big subfield of AI is machine learning (ML) [31, 32] that has its theoretical foundation in statistical learning theory [33, 34]. ML deals with algorithms that improve on a certain task with increasing amount of knowledge (i.e., data) and make predictions *"without being explicitly programmed"* [35, 36]. More formally, Tom M. Mitchell defined ML algorithms the following way [37]: *"A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P if its performance at tasks in T, as measured by P, improves with experience E."* The learning process can be summarized as a combination of representation, evaluation and optimization [38]: First the problem must be expressed in a form that is understandable to a computer. Specifically, one needs to define the space of possible functions or hypotheses it can learn and – related to that – the input representation. Then, an evaluation function (also called objective) needs to be defined that allows to differentiate between well- and poorly performing models. Finally, the optimization method has to be chosen, which allows to search for the model with highest score.

Usually, ML is further divided into the areas of supervised learning, unsupervised learning, and reinforcement learning (cf. Fig. 1.1b), which will be explained in the following. We rely on supervised an unsupervised learning techniques in this thesis.

A supervised ML method aims at learning the functional dependence between inputs $\mathbf{x}$ and outputs $\mathbf{y}$ given a set of training examples. The inputs may be seen as vectors of a certain dimension, whose components are called features. As we will discuss in section 1.4, when applying ML in a supervised setting to materials science problems and generally in physics, including a-priori knowledge such as physically relevant symmetries can drastically improve the performance of ML models. The development of a suitable, physically motivated input representation, the *descriptor*, is of major importance [39]. The input $\mathbf{x}$ is not constrained to take vectorial form but can also correspond, for instance, to the pixels of an image or a graph representing molecules or crystals [40, 41]. Depending on the definition of the outputs, supervised ML is further subdivided into regression (outputs are real numbers) and classification problems (outputs correspond to a particular number of classes). A more formal description can be found in section 2.2.

In unsupervised learning, the inputs are not labeled like in a supervised setting. The goal is to find hidden patterns in the data, given the way it is represented by the features. Clustering is one example of an unsupervised ML technique. Here, the data points are divided into groups or clusters. The grouping is meaningful in the sense that points assigned to the same cluster are more similar to each other than to points outside the cluster. This establishes a notion of similarity that may be related to a physical concept, thus revealing hidden regularities in the data. Unsupervised learning can also be used to reduce the dimensionality of the input feature space, allowing data visualizations in a compressed and more comprehensible representation, e.g., in two-dimensional (2D) Euclidean space. The dimensionality-reduced description may be used to boost model performance – since redundant features are removed – or to reveal relations and similarities between data points. Both clustering and dimensionality reduction is employed in this work, in particular to *explain* the employed ML model (section 6.2.2) and to perform exploratory analysis (section 6.6.2). Note that explainability is part of the controversially discussed topic of *interpretability* in ML, i.e., how predictions of ML models, which can be complex and black-box like, can or should be justified [42–44].

A further subfield of ML, reinforcement learning [45], has strongly influenced the public perception of AI. In particular, the combination of reinforcement learning with tree search and deep learning has lead to breakthroughs in game playing [18, 19, 46–48]. This has raised significant attention, also in materials science [49–51]. In reinforcement learning, a so-called agent is interacting with the environment by taking a (time) sequence of actions. At each step of the sequence, an action is chosen based on the observed state of the environment. This action is applied to change the environment state and a reward is calculated and transmitted to the agent. Then, the agent receives a new observation of the environment state and chooses a new action etc. (cf. chapter 1.3.4 of [52]). The goal of reinforcement learning is to find a good policy, i.e., a function that maps environment observations to actions and thus determines the agent's behavior.

A highly relevant subclass of ML algorithms does not only learn the predictive model but also representations of the data, starting from some initial input encoding. Deep learning [14], which is a cornerstone of this work, and symbolic regression [53–55] are prominent examples. Deep neural networks, like other ML algorithms, learn from experience. Notably, they extract a hierarchy of concepts, where each concept is built out of simpler ones. This allows to learn highly complex relationships. Depicting the extracted hierarchy as a graph, one would obtain a deep graph with many layers – which is why this field is called deep learning [14].

Finally, one can distinguish AI methods that perform either exploratory or confirmatory analysis [56]. The latter tools learn and predict from data as it is the case in supervised ML. In an exploratory setting, the data is inspected via AI to infer regularities or rules that lead not only to insights but also new questions, which itself lead to further studies. For example, applying clustering (i.e., unsupervised learning) to a given dataset yields groups of similar points. Based on this, one can conduct additional tests to find out if there is a physically meaningful reason for the discovered grouping. In this thesis, we conduct exploratory analysis using both unsupervised clustering methods and a supervised Bayesian neural network, a sub-family of artificial neural networks (ANNs), cf. section 6.6.2.

1.2 Deep learning for materials science

In the following, we present an extraction of deep-learning related research in materials science. More detailed reviews can be found in [21, 57, 58] and [59–61]. Training deep neural networks typically requires an amount of data that is not always available in materials science problems, in particular if time-consuming calculations are involved, e.g., density-functional theory or molecular-dynamics simulations. Thus there is a substantial amount of effort devoted to other, less data-demanding ML algorithms such as decision trees [40, 62, 63], kernel ridge regression [64–66], subgroup discovery [67], and symbolic regression [54, 55].

Already before ANNs had spiked attention, they have been employed to learn potential energy surfaces: inspired by the pioneering work [68], ANNs have been applied to this task in [69] and further developed since then [70–72]. ANNs can also be used to represent the quantum many-body wave function, where the initial approach focused on discrete spin-lattice systems [73]. This has been generalized to bosons in real space [74, 75] and even electrons in real space [76]. Further improvements have recently been made with the so-called PauliNet [77] and FermiNet [78] architectures. The highly complex and vital task of protein structure prediction, specifically protein folding, has been substantially improved using ANNs [79]. Closely related to this thesis, Ziletti *et al.* employed a convolutional neural network [80] for robust and interpretable crystal-structure classification, given Cartesian coordinates labeled by chemical-species symbols [26]. For the analysis of high-resolution microscopy data (in particular electron microscopy), Kalinin and coworkers have introduced several DL frameworks [81–84]. For instance, *AtomNet* [81], allows to recognize atomic positions from noisy scanning transmission electron microscopy (STEM) images (which is employed in sections 5.2 and 6.5). The prediction of inorganic compound properties (such as formation energy, volume per atom etc.) has been addressed with different input information (i.e., descriptors) and neural network architectures: The *ElemNet* architecture [85,86] or *CRYSPNet* [87] use only chemical composition as input, while others use geometrical information. For instance, variants of message-passing neural networks [88] such as *Schnet* [89,90] can be employed for property prediction. These frameworks can also be used to predict potential energy surfaces and force fields. Related to this, crystal-graph convolutional neural networks [41] are based on a crystal graph representation [40], demonstrating strong performance in materials property prediction, while the internal representations have also been analyzed using unsupervised techniques [91]. *Euclidean neural networks* [92] have been developed for the analysis of three-dimensional (3D) geometries in a rotation-, translation-, and permutation-equivariant fashion. Besides predictive models, generative models have been applied in materials science as well (i.e., networks that generate inputs $\mathbf{x}$ from labels $\mathbf{y}$ instead of taking the inverse, predictive route). For instance, generative neural networks have been used for drug design [93], also in combination with reinforcement learning [49], as well as the sampling of equilibrium states of many-body systems [94] or to learn embeddings from molecular simulations data [95].

1.3 Crystal-structure recognition

The goal of crystal-structure identification is to assign a symmetry label (for instance, the space group) to a given atomic structure (cf. Fig. 1.2). More generally, one wants to find the most similar structure within a list of given known systems – the *structural classes* (cf. section 4.3.3.2), identified by stoichiometry, space group, number of atoms in the unit cell, and location of the atoms in the unit cell (the Wyckoff positions). This can help to understand important physical properties such as the hardness of industrial steel [96]. In this context, the importance of grain boundaries has been studied in numerous experiments [97], also in combination with crystal-structure prediction [98]. Beyond bulk materials, two- (2D) and one-dimensional (1D) systems have far-reaching technological applications, such as solar energy storage, DNA sequencing, cancer therapy, or even space exploration [99, 100].

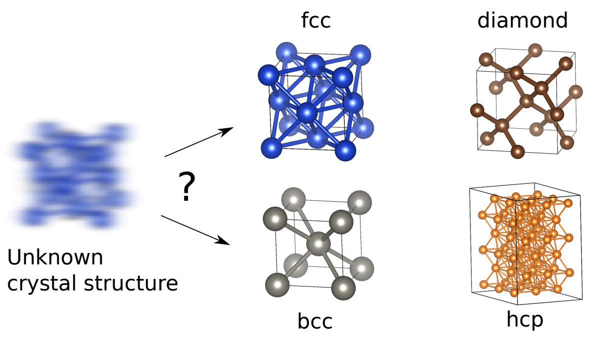

Figure 1.2: Schematic overview of the general crystal-structure recognition problem: assign the most similar crystal prototype to a given unknown and possibly highly defective atomic structure. Here, the set of reference structures to which the input system is compared to are the face-centered cubic (fcc), body-centered cubic (bcc), diamond, and hexagonal close-packed (hcp) structures.

Methods for automatic crystal-structure recognition are required to analyze the continuously growing amount of geometrical information on crystal structures, from both experimental and computational studies. Millions of crystal structures alongside calculated properties are available in large computational databases such as the NOvel MAterials Discovery (NOMAD) Laboratory [23, 24], AFLOW [101, 102], the Open Quantum Materials Database (OQMD) [103], Materials Project [104], or repositories specialized in 2D materials [105,106]. In STEM [107], atomic positions can be reconstructed from atomic-resolution images for specific systems, e.g., the 2D material graphene [81]. Three-dimensional atomic positions are provided by atom probe tomography (APT) [108] and atomic electron tomography (AET) experiments [109]. Still, substantial levels of noise due to experimental limitations and reconstruction errors are present in atomic positions, e.g., distortions beyond the physical level or, in case of APT, large amount of missing atoms (at least 20%, due to the limited detector efficiency [110]). Crystal-structure recognition schemes should be able to classify a large number of structural classes (also beyond bulk materials) while at the same time being robust with respect to theoretical or experimental sources of inaccuracy and physically driven deviations from ideal crystal symmetry (e.g., vacancies or thermal vibrations). Given the large amount of data, the classification should be fully automatic and independent from the manual selection of tolerance parameters (which quantify the deviation from an ideal reference structure).

Current computational methods for crystal-structure recognition are based either on space-group symmetry or local topology. For space-group-based approaches (notable examples being Spglib [111] and AFLOW-SYM [112]), the allowed symmetry operations are calculated directly from the atomic positions to infer a space group label. For local-topology-based approaches, the local atomic neighborhood of each individual atom is classified into a prepared list of reference structures. This assignment may coincide with the space-group-based classification. Examples of these methods are common neighbor analysis [113], adaptive common neighbor analysis [114], bond angle analysis [115], and polyhedral template matching [116]. Concerning classification performance, space-group approaches can treat all space groups but are limited with respect to noise, while local-topology methods can be quite robust but only treat a handful of classes. Considering big data, a further limitation of space-group-based methods such as spglib is threshold dependence, as the possible symmetry operations are determined based on user-specified tolerances for positions and angles. Above a certain noise level, no symmetry beyond the identity operation (space group P1) can be identified, regardless of the chosen threshold. This happens already for low noise levels, e.g., removing only a few atoms, as shown in section 5.1. Moreover, none of the available structure recognition schemes can recognize more complex nanostructures such as nanotubes.

To improve on currently available crystal-structure identification methods, we can benefit from advances in ML. In particular, artificial neural networks used in deep learning [12, 14] have outperformed state-of-the-art machine-learning models in numerous applications. The distinguishing feature of neural networks is the ability to learn complex representation directly from data; these representations become more abstract the deeper the network , i.e., the more layers it has. Due to their large model complexity, deep neural networks typically require large amount of data to be trained. They are also hard to interpret as learning is performed in a distributed way with the information spread across millions of parameters (the so-called network weights). Still, deep learning shows compelling success in problem settings whose dimensionality exceeds the model complexity by far [117]. To understand this statement, let us consider the binary classification example of distinguishing grayscale pictures of dogs and cats. For images with 100×100 pixels, each pixel taking one out of 256 values, the space of possible images is $256^{100 \cdot 100}$. Yet, networks with millions and thus comparatively tiny amount of parameters can successfully solve this task. Similarly, the space of possible crystalline solids is practically infinite and still ML methods can be applied to predict physical properties, e.g., using neural networks [41]. Note that ML has already been applied to crystal-structure recognition [118–121]. However, none of these works were able to go beyond a handful of classes and show high robustness at the same time. We need to extend previous work [26] which employed a convolutional neural network to distinguish crystal structures based on a novel diffraction fingerprint. While being extremely robust and providing interpretable predictions, this approach is restricted to elemental solids and a limited number of space groups.

Here, we propose a robust, threshold-independent crystal-structure recognition framework (ARtificial-Intelligence-based Structure Evaluation, short *ARISE*) [122] to classify a diverse set of 108 structural classes, comprising bulk, 2D, and 1D materials. Bayesian neural networks [123, 124] are used, i.e., a recently developed family of neural networks that yields not only a classification but also uncertainty estimates. These estimates are principled in the sense that they approximate those of a well-known probabilistic model (the *Gaussian process*). This allows to quantify prediction uncertainty, but also the degree of crystalline order in a material. ARISE performance is compared with the current state of the art, and then applied to various computational and experimental atomic structures. Crystal characterization and identification of hidden patterns is performed using supervised learning (ARISE) as well as the unsupervised analysis (via clustering and dimensionality reduction) of the internal representations of ARISE.

1.4 The importance of descriptors

To apply ML to high-dimensional condensed matter and materials science problems, finding a suitable mapping of the input onto a descriptor is crucial [39]. In particular, one needs to ensure that physical requirements we know to be true (e.g., conservation laws, translational or rotational invariance) are respected by construction.

In our setting of single-crystal classification, the starting point is a crystal structure, represented by atomic positions and chemical-species symbols; the goal is to assign to this structure its most similar crystallographic prototype. A-priori knowledge based on physical requirements can greatly increase the quality of machine-learning models, and in particular their generalization ability (i.e., the capability of models to not simply memorize the training data but also perform well on unseen examples). Let us take the example of rotational invariance: two crystal structures that differ only by a rotation must have the same classification label. Clearly, atomic coordinates in real space are not rotationally invariant, and thus are not a good representation (i.e., input) for machine-learning models. As an attempt to fix this, one might include a discrete subset of orientations in the training set, hoping that the model will generalize to unseen rotations. However, there is no guarantee that the model will learn the rotational symmetry, and if it does not, it will fail to generalize and return different predictions for symmetrically equivalent structures. In contrast, when a rotationally invariant representation is used, only one crystal orientation needs to be included in the training set and the model will generalize to all rotations by construction. This reasoning readily applies to other physics requirements such as translational invariance, or permutation invariance (for atoms with same chemical species).

Some of the most well-known descriptors in physics and materials science incorporate these physical invariants: symmetry functions [125], the smooth-overlap-of-atomic-positions descriptor (SOAP) [126, 127], the many-body tensor representation [128], and the moment tensor potential representation [129]. We choose SOAP as materials representation, which is a state-of-the-art descriptor for representing chemical environments that is invariant with respect to translations, rotations and permutations of identical atoms. SOAP has been successfully applied to numerous materials science problems: interatomic potentials fitting [130], structural-similarity quantification using kernel functions [131], prediction of material properties (for instance, formation enthalpy [132]) or grain-boundary characteristics such as energy and mobility [133]. In this work, we construct the SOAP descriptor such that a crystal structure is represented by a vector of fixed length, independently of the number of atoms and chemical species (see 4.3.2 for more details). Note that any other suitable descriptors (i.e., a descriptor respecting above-mentioned physical requirements) can be used as input for our procedure. In particular, we provide the code framework *ai4materials* (https://github.com/angeloziletti/ai4materials) into which alternative representations can be readily integrated.

Chapter 2

Deep learning

This chapter provides an introduction to deep learning. We mainly follow the two valuable resources [14] and [52], where for the latter we refer to the chapters of the online version (https://d2l.ai/index.html) while an arXiv preprint (with different section numbering) can be found at [134]. Section 2.1 explains a standard ANN architecture – fully connected neural networks. The most simple form, the so-called perceptron, is described in section 2.1.1 and extended to multilayer perceptrons in section 2.1.2. Training and optimization of deep neural networks is reviewed in section 2.2, where loss functions are discussed (in particular those being relevant for classification tasks, cf. 2.2.1) as well as different strategies for optimization (cf. 2.2.2) and regularization (cf. 2.2.3). Tuning of model hyperparameters is explained in section 2.2.4, in particular Bayesian optimization strategies. Then, section 2.3 discusses Bayesian deep learning, where first the fundamentals of Bayesian modeling in context of ANNs are explained, including a short recap of so-called variational inference (cf. section 2.3.1), a method that can be employed to perform approximate Bayesian inference. Section 2.3.2 discusses Monte Carlo dropout, which is a practical and principled way to obtain uncertainty estimates from deep neural networks. Online tutorials that complement sections 2.1 and 2.2 may be read in parallel and have been published at the NOMAD analytics toolkit (https://nomad-lab.eu/aitoolkit). Alternatively, they can be found on github via the URLs https://github.com/AndreasLeitherer/Tutorial_multilayer_perceptron and https://github.com/AndreasLeitherer/Tutorial_CNN. For Bayesian deep learning (complementing section 2.3), we uploaded a tutorial on ARISE, which is based on Bayesian neural networks, to the NOMAD analytics toolkit (see https://analytics-toolkit.nomad-coe.eu/tutorial-ARISE or on github https://github.com/AndreasLeitherer/Tutorial_ARISE).

2.1 Fully connected neural networks

2.1.1 The perceptron

The origin of ANNs dates back to the early 1940's. The most simple form of an ANN is the perceptron, which was developed by Frank Rosenblatt in 1957 [135] and is biologically motivated (see the simplifying sketch of a biological neuron in Fig. 2.1a). The output of a perceptron is computed by first linearly combining the input features and then applying

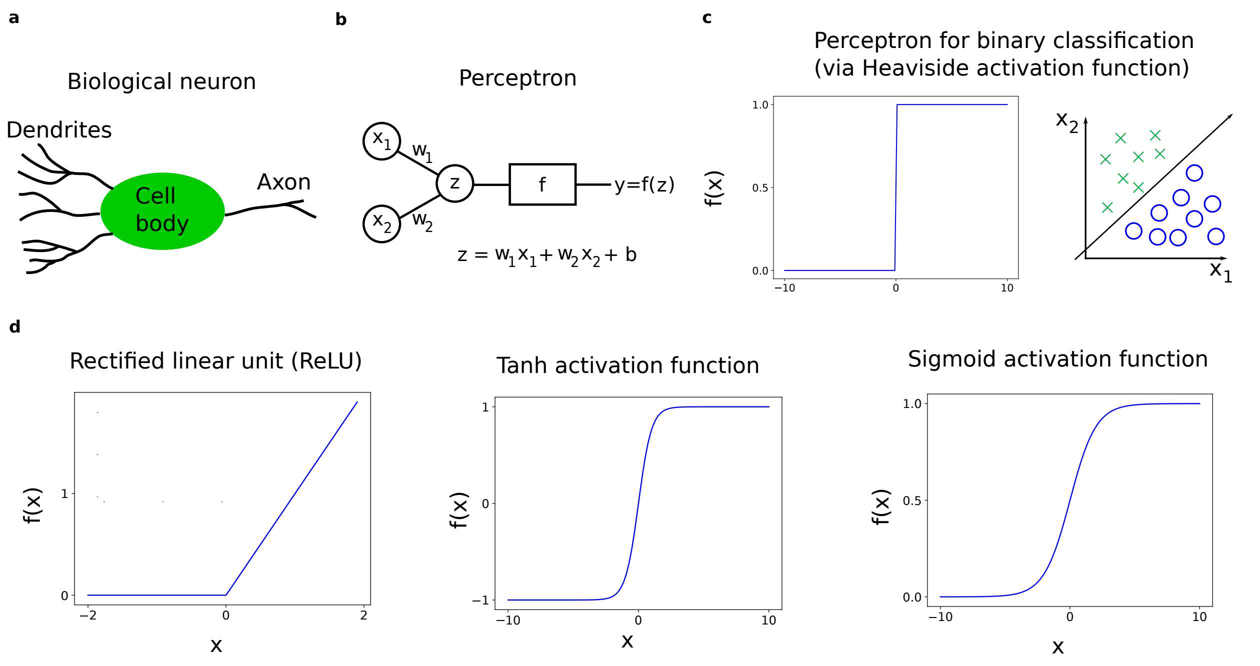
a
Biological neuron
Dendrites
Cell body
Axon
b
Perceptron
x1
x2
w1
w2
z
f
y=f(z)
z = w1x1+ w2x2+ b
c
Perceptron for binary classification (via Heaviside activation function)
f(x)
x2
x1
d
Rectified linear unit (ReLU)
Tanh activation function
Sigmoid activation function
f(x)
x

Figure 2.1: Introduction of the perceptron. **a** Sketch of a biological neuron. **b** Illustration of a simple perceptron. **c** Heaviside activation function (left) employed in a simple binary classification task (right). **d** Three popular activation functions.

a non-linear function (the so-called activation function). More formally, the output y of a perceptron is computed as

$$y = f(z) := f\left(\sum_{i=1}^{N} w_i x_i + b\right), \tag{2.1.1.1}$$

where z denotes the linear combination of input features $x_i, i = 1, ..., N$, $w_1, w_2, ..., w_N$ the weights, b the bias, and f the activation function. An example for $N = 2$ is shown in Fig. 2.1b.

As a first example, let us consider a binary classification task in which two classes, marked by circles and crosses in the right part of Fig. 2.1c, ought to be distinguished using two input features x_1, x_2. This could represent a medical problem setting, where one wants to predict if a patient has a certain disease based on two factors (e.g., age and height). As activation function, the Heaviside function is employed, for which the model output is either 1 for $z > 0$ or 0 otherwise. The possible output values correspond to the two classes. Given fixed weights w_1, w_2, the bias term b defines when the neuron is firing ($f(z) = 1$) or not ($f(z) = 0$) for given input features x_1, x_2. Training of a perceptron can be perceived as changing the model parameters such that the optimal position of a straight line is found, which serves as a decision boundary between the two classes (cf. Fig. 2.1c). We will discuss the training and optimization procedure for modern neural networks in more detail in section 2.2. Note that the Heaviside activation function is not used in modern deep-learning applications but rather the Rectified Linear Unit (ReLU) or the sigmoid and hyperbolic-tanges functions (see Fig. 2.1d). Employing the sigmoid function would yield a logistic regression problem (see section 4.3 of [34])

The use of non-linear functions is essential: if no activation function was used (i.e., the identity or linear activation function – reducing Eq. 2.1.1.1 to a linear regression problem), the class of possible functions that the model can represent would be drastically reduced.

2.1.2 Multilayer perceptrons

Extending the idea of simple perceptrons, multilayer perceptrons (MLPs) are constructed as a sequence of layers (cf. Fig. 2.2a). Each layer consists of a predefined number of neurons, where the neurons of the first layer (the input layer) correspond to the input features $\mathbf{x} = (x_1, x_2, ..., x_d)^T \in \mathbb{R}^d$. The subsequent layers are called hidden layers and the final neurons the output layer. The MLP shown in Fig. 2.2a has four output neurons, each corresponding to a distinct class, i.e., we consider a multi-class classification problem. Hidden and output neurons are a combination of all neurons from the previous layer (i.e., a fully connected neural network) which is indicated by the connections in Fig. 2.2a. For instance, the so-called activation $a_1 \in \mathbb{R}$ highlighted in Fig. 2.2b (for $\mathbf{x} \in \mathbb{R}^6$) is computed as

$$a_1 = f(z_1) = f(w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4 + w_5x_5 + w_6x_6 + b_1). \tag{2.1.2.1}$$

Here, $w_i \in \mathbb{R}, i = 1, ..., 6$ are the weights and $b_1 \in \mathbb{R}$ denotes the bias value. These constitute the model parameters that are linearly combined with the input $\mathbf{x}$ to yield z_1. Finally, the activation function f is applied, resulting in the activation value a_1. This process, which is called forward propagation, is repeated for all neurons and layers until the output $\mathbf{o} = (o_1, o_2, ..., o_N)$ is obtained. Since there are no cycles in the connections, these networks are called feedforward neural networks. In so-called recurrent neural networks this condition is relaxed, leading to breakthrough results in natural language processing [8, 136].

Already at this level, one of the main characteristics of multilayer neural networks becomes clear: the ability to learn new representations [138]. This distinguishes neural networks from other ML algorithms (e.g., decision trees). The extracted representations become more abstract the deeper the network is, i.e., the more hidden layers it has. For instance, the input features may correspond to the pixels of an image that needs to be classified into a certain number of categories (e.g., if it contains a dog or a cat). In the first layers, the network may extract features such as lines, while in later layers collection of edges and larger parts can be detected. Note that the architecture shown in Fig. 2.2a has only one hidden layer and thus would be called a shallow neural network. Models with multiple hidden layers are called deep neural networks (cf. Fig. 2.2d).

To formalize forward propagation for an arbitrarily sized MLP, we consider the model in Fig. 2.2a. The activation vector $\mathbf{a} \in \mathbb{R}^5$ is defined as

$$\mathbf{a} = f(\mathbf{A}\mathbf{x} + \mathbf{b}). \tag{2.1.2.2}$$

This is essentially an affine transformation (defined by a 5×6 matrix $\mathbf{A}$ and bias vector $\mathbf{b} \in \mathbb{R}^5$) followed by element-wise application of the (non-linear) activation function f. The weights of the linear combinations are collected in the matrix $\mathbf{A}$ and the offsets in the bias vector. The output activations $\mathbf{o}$ are obtained by applying a further affine transformation

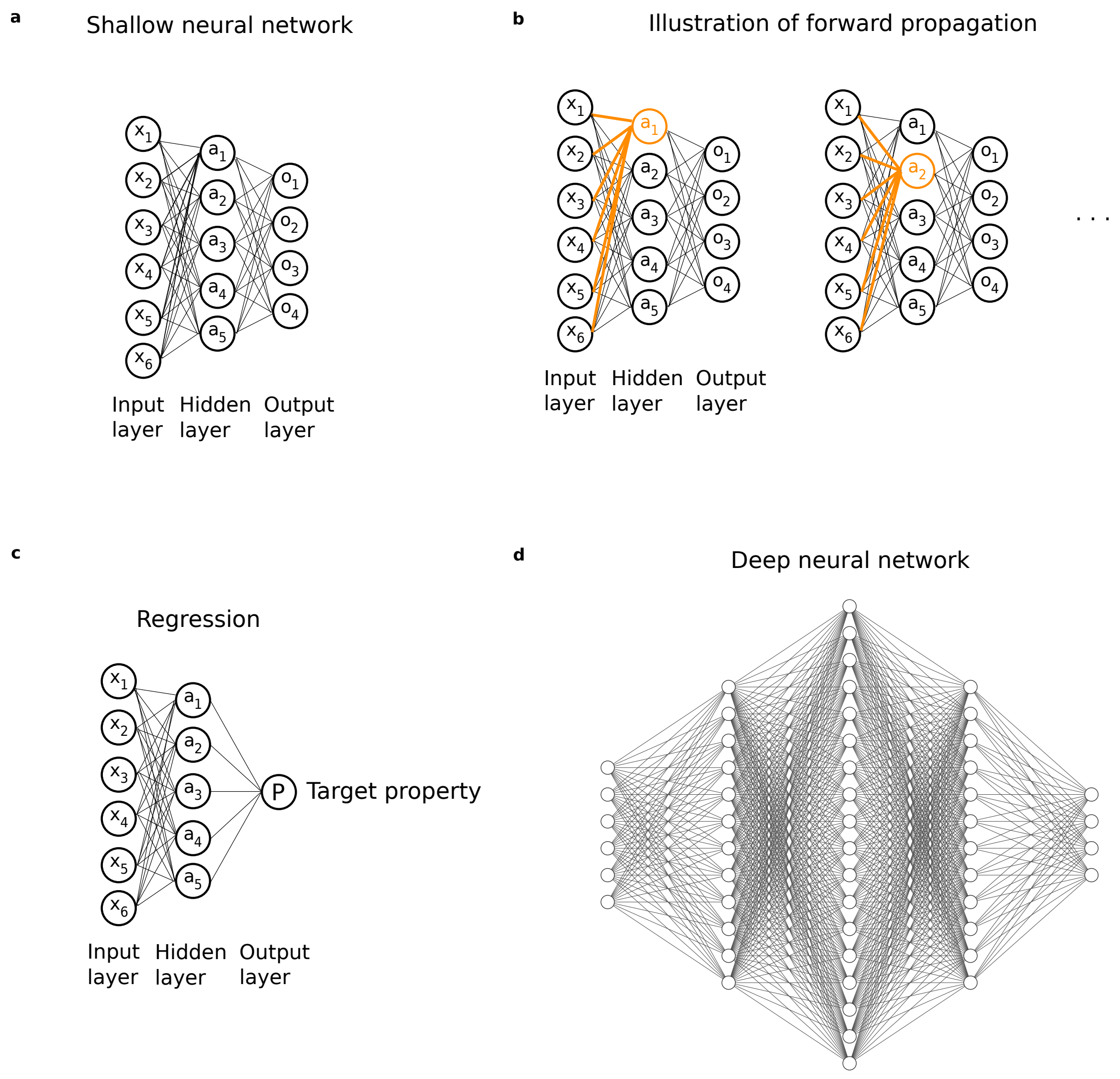

Figure 2.2: Introduction of fully connected neural networks (multilayer perceptrons). **a** Shallow neural network with one hidden layer. The four output neurons may correspond to four different classes (e.g., structural classes in a crystal-structure classification problem). **b** Illustration of forward propagation for the first two neurons in the hidden layer. **c** Adaption of the architecture of **a**,**b** for a regression task, i.e., the prediction of a scalar target property P. **d** Example of a multilayer perceptron with three hidden layers, qualifying as a deep neural network. Modern architectures can reach more than 50 layers [137].

(matrix $\mathbf{A}'$, bias $\mathbf{b}'$) and activation function f', which yields

$$\mathbf{o} = f'(\mathbf{A}'\mathbf{a} + \mathbf{b}'). \tag{2.1.2.3}$$

The final activation function f' is chosen in a specific way, usually depending on the task being either regression or classification. We will address this in the last two paragraphs of this section. To simplify the above expression for $\mathbf{o}$, it is common to change the definition of input vector and weight matrices such that the bias terms are incorporated into the mapping matrix $\mathbf{A}$. To illustrate this, we consider the simplified case of two input features and two activations $a_1 = w_{11}x_1 + w_{12}x_2 + b_1$ and $a_2 = w_{21}x_1 + w_{22}x_2 + b_2$. Then, $\mathbf{A}$ and $\mathbf{b}$ are defined as

$$\mathbf{A} = \begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \end{bmatrix}, \mathbf{b} = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}. \tag{2.1.2.4}$$

Introducing the new definitions

$$\mathbf{W} = \begin{bmatrix} b_1 & w_{11} & w_{12} \\ b_2 & w_{21} & w_{22} \end{bmatrix}, \mathbf{x} = \begin{bmatrix} 1 \\ x_1 \\ x_2 \end{bmatrix}, \tag{2.1.2.5}$$

allows to incorporate the bias term and replace $\mathbf{A}\mathbf{x} + \mathbf{b}$ with $\mathbf{W}\mathbf{x}$. Similarly, we introduce weight matrices $\mathbf{W}$, $\mathbf{W}'$ for the MLP shown in Fig. 2.2a, which yields a compact expression for the output given by

$$\mathbf{o} = f'(\mathbf{W}'\mathbf{a}) = f'(\mathbf{W}'f(\mathbf{W}\mathbf{x})). \tag{2.1.2.6}$$

This equation makes it more clear that a multilayer perceptron is a concatenation of affine and non-linear transformations, parametrized by a set of weights and biases which are optimized to fit the task at hand. Eq. 2.1.2.6 is straightforward to generalize, which yields

$$\mathbf{o} = f^N(\mathbf{W}^N f^{N-1}(\mathbf{W}^{N-1} ... f^1(\mathbf{W}^1\mathbf{x})). \tag{2.1.2.7}$$

An example of the case of $N = 3$ is shown in Fig. 2.2d, which is also called a 3-layer neural network following the convention that only hidden layers count.

The class of functions a MLP can represent would be strongly diminished if linear instead of non-linear activation functions are employed. In particular, the *universal approximation theorem* guarantees that using a feed-forward neural network with at least one hidden layer and any activation function (such as sigmoid), one can approximate any continuous function with arbitrary precision – as long as the hidden layer contains sufficiently many neurons [139, 140]. The universal approximation theorem has initially been proven for activation functions that converge to a constant for inputs tending to $\pm\infty$ while extensions cover also activation functions that are usually employed today (such as the ReLU function [141]). Notably, the universal approximation theorem does not provide guidance for architecture choices (number of layers, neurons etc.). Moreover, no generalization guarantees are provided, i.e., how model performance can be improved for data points that have not been included during training.

As discussed in context of Eq. 2.1.2.3, the final activation function is chosen depending on the application. In a classification setting, the so-called softmax function is employed as activation function for the final layer. This allows to obtain predictions $\hat{y}_i := o_i$ whose sum is normalized to one such that each y_i can be interpreted as the classification probability of class i. Denoting the linear combination of activations from the previous layer as $\mathbf{z}$ (cf. Eq. 2.1.2.1), the final predicted value can be obtained, using the softmax function, via

$$\hat{y}_i := o_i = [f_{\mathrm{softmax}}(\mathbf{z})]_i = \frac{\exp(z_i)}{\sum_k \exp(z_k)}. \tag{2.1.2.8}$$

The predicted class label is inferred by selecting the most likely class, i.e., one computes

$$\underset{j}{\operatorname{argmax}}\ \hat{y}_j. \tag{2.1.2.9}$$

Let us consider the case of a simplified crystal-structure classification sketched in Fig. 1.2, where the task is to assign the most similar crystal structure (out of a pool of reference systems) to a given unknown crystal structure. Employing the multilayer perceptron architecture shown in Fig. 2.2a, each of the four output neurons correspond to a specific crystal structure. Using the softmax function for the normalization of the output neurons, one can interpret $\mathbf{y} = (1, 0, 0, 0)$ as the prediction of fcc symmetry with 100% probability. Similarly, (0,1,0,0) may be associated with bcc symmetry etc. This is called one-hot encoding and corresponds to representing a given number of N classes via the standard basis in $\mathbb{R}^N$, i.e., by N vectors $\mathbf{e}_i = (0, ...0, 1, 0, ..., 0), i = 1, ..., N$, where all components of e_i are zero except for the ith entry. Instead of one-hot encoding one may also encode the classes as integers (i.e., predict a single integer that corresponds to a specific class). Since this introduces a fictitious order between the classes, one-hot encoding is preferred. For regression problems, only one output neuron is used (cf. Fig. 2.2c) and the output activation may be chosen depending on the desired range of the target (cf. Fig. 2.1d).

2.2 Training and optimization

To establish a deep-learning method, one needs to specify dataset, model architecture, loss function, and optimization protocol. While neural networks can be employed in an unsupervised setting (cf. chapter 14 of [14]), the supervised-learning scenario is typically considered. Here, the goal is to learn the functional relationship between inputs $\mathbf{x}$ and outputs $\mathbf{y}$ given a set of m training examples $D = \{(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})_{i=1}^{m}\}$. A typical assumption is that the training points are drawn independently from identical distributions, the so-called i.i.d. assumption. This implies that the data is sampled without memory. Regarding the model, we employ a MLP in this thesis which is essentially a non-linear function $f_{\boldsymbol{\omega}}$, parametrized by a set of weight matrices $\boldsymbol{\omega} := \{\mathbf{W}_i\}_{i=1,...,N}$. For a given input $\mathbf{x}$, the predictions are obtained via $\hat{\mathbf{y}} = f_{\boldsymbol{\omega}}(\mathbf{x})$. The optimal set of parameters is found by minimizing a cost or loss function $J(\boldsymbol{\omega})$ that quantifies the difference between ground truth and predictions. The loss function is minimized using gradient-descent based algorithms (cf. section 2.2.2). Typically one obtains a collection of models, obtained from several optimization iterations, in which a different set of hyperparameters is tested, for instance, the number of layers and neurons in each layer (cf. section 2.2.4). The selection of the optimal model is based on a suitable measure, for instance, the classification accuracy in our case of crystal-structure identification. This performance measure is only optimized indirectly, via minimizing the cost function. This distinguishes a ML problem from a pure optimization setting, in which minimization of the loss would be the only goal.

2.2.1 Loss functions

Ideally, one would like to minimize [14]

$$J(\boldsymbol{\omega}) = \mathbb{E}_{\mathbf{x},\mathbf{y}\sim p_{\text{data}}} L(f_{\boldsymbol{\omega}}(\mathbf{x}), \mathbf{y}), \tag{2.2.1.1}$$

where $L(f_{\boldsymbol{\omega}}(\mathbf{x}), \mathbf{y})$ quantifies the loss for a single data point and p_{data} is the data-generating distribution which $\mathbf{x}, \mathbf{y}$ follow (denoted as $\mathbf{x}, \mathbf{y} \sim p_{\text{data}}$). Unfortunately, one typically cannot access this complex distribution, especially in materials science where the number of possible materials is practically infinite. Thus, one minimizes the so-called empirical risk

$$\hat{J}(\boldsymbol{\omega}) = \frac{1}{m}\sum_{i=1}^{m} L(f_{\boldsymbol{\omega}}(\mathbf{x}^{(i)}), \mathbf{y}^{(i)}), \tag{2.2.1.2}$$

i.e., the goal is to find the optimal parameters

$$\hat{\boldsymbol{\omega}} = \underset{\boldsymbol{\omega}}{\operatorname{argmin}}\ \hat{J}(\boldsymbol{\omega}) = \underset{\boldsymbol{\omega}}{\operatorname{argmin}}\ \frac{1}{m}\sum_{i=1}^{m} L(f_{\boldsymbol{\omega}}(\mathbf{x}^{(i)}), \mathbf{y}^{(i)}). \tag{2.2.1.3}$$

To obtain the model that generalizes best to unseen data, one typically splits the initial data set D into training, validation and test set, denoted as $D_{\text{train}}, D_{\text{val}}$, and D_{test}, respectively. The model is trained on D_{train}, while the generalization error is estimated using D_{val}. In particular, one can optimize model-specific hyperparameters using the validation set. In case of deep neural networks, this may be the number of layers or the neurons in each layer and one may select the architecture with the optimal performance measure on D_{val}. Other strategies such as cross validation employ different splits during training (cf. section 7.10 of [33]). Specifically, k-fold cross validation divides the training set into k groups (after splitting off a separate test set). Each group is considered as a hold-out test (or rather validation) set while fitting the model to the remaining data points of the training set. This yields k models trained on different splits whose performance measures may be averaged, providing an improved measure of generalization ability. Due to the increased computational cost, especially in case of deep learning, cross validation strategies are often avoided. Once a satisfactory model is found, the performance measure is evaluated on D_{test}.

For classification, a common loss function is the so-called cross-entropy loss which is defined as [52]

$$L_{\text{cross-entropy}}(f_{\boldsymbol{\omega}}(\mathbf{x}^{(i)}), \mathbf{y}^{(i)}) := -\sum_{j=1}^{C} \mathbf{y}_j^{(i)} \log \left[f_{\boldsymbol{\omega}}(\mathbf{x}^{(i)})\right]_j, \tag{2.2.1.4}$$

where C denotes the number of classes. This equation can be derived using the principle of maximum likelihood as well as the i.i.d assumption. We refer to section 5.5 of [14] as well as sections 18.7 and 18.11.5 of [52] for more details.

For regression, mean squared error is a popular loss function [14], which is defined as

$$L_{\text{mean-squared-error}}(f_{\boldsymbol{\omega}}(\mathbf{x}^{(i)}), \mathbf{y}^{(i)}) := ||f_{\boldsymbol{\omega}}(\mathbf{x}^{(i)}) - \mathbf{y}^{(i)}||_2^2, \tag{2.2.1.5}$$

where $||.||_2$ is the Euclidean norm.

2.2.2 Optimization strategies

This section overviews some of the most important optimization techniques that are employed in deep learning. This includes the *Adam* optimization algorithm that is used in this thesis. We mainly follow section 8 of [14] and section 11 of [52].

Neural networks are essentially highly non-linear functions which is why the optimization of a cost function is a non-convex problem. This comes with several pitfalls such as multiple local minima that are far away from the global minimum and in which the optimization algorithm might get stuck. Further possible problems are saddle points and vanishing gradients (cf. section 11 in [52]), the latter being more frequently encountered in recurrent neural networks [8, 136]. Fortunately, several methods have been developed to address these problems.

Typical optimization strategies in deep learning rely on the gradient-descent algorithm. This method can be used to minimize a scalar function $f : \mathbb{R}^d \to \mathbb{R}$, representing the cost function. Specifically, the cost is minimized by iteratively moving the model parameters $\boldsymbol{\omega}$ into the direction of the negative gradient via the update rule [14]

$$\boldsymbol{\omega} \leftarrow \boldsymbol{\omega} - \eta \nabla \hat{J}(\boldsymbol{\omega}), \tag{2.2.2.1}$$

where η denotes the learning rate. In simple gradient descent, η is fixed for all iterations and influences the step size, i.e., the pace at which the minimum is approached. A large learning rate might lead to divergence (i.e., a steady value of the loss function that may correspond to a local minimum will never be reached), while a tiny value will lead to long convergence times. The backpropagation algorithm [8] provides a computationally efficient way to compute the gradient in Eq. 2.2.2.1.

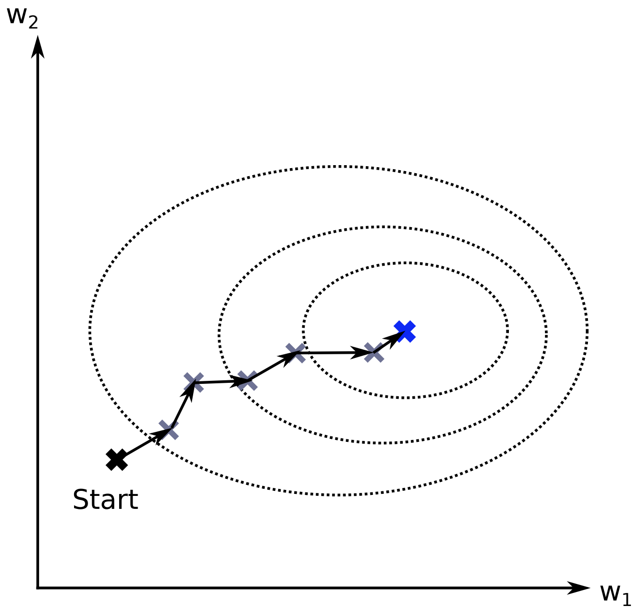

Figure 2.3: Illustration of gradient descent applied to the minimization of a loss function that is defined for a model with parameters w_1, w_2. The steps from start (black cross) to the global minimum (blue cross) via intermediate steps (gray crosses) are shown.

Using Eq. 2.2.2.1 as optimization protocol, which is also called batch gradient descent, can be computationally costly. The gradient of the empirical loss has to be computed, which involves a sum over all training points. In ML and in particular in deep learning, this calculation is computationally expensive, since large training sets are often required for a satisfactory generalization performance.

As an alternative to batch gradient descent, one can randomly select subsets of points for the gradient computation in Eq. 2.2.2.1. In stochastic gradient descent (SGD), only one randomly selected data point is used to approximate the gradient of the loss function. This leads to a more noisy trajectory towards the minimum of the loss landscape. On the positive side, this can help to avoid that the algorithm is trapped in local minima. On the downside, this strategy may prolong or even prohibit convergence. Notably, SGD is computationally inefficient since the advantages of vectorization cannot be exploited on Central Processing Units (CPUs) and Graphics Processing Units (GPUs), see also section 11.5.1 of [52]. That is why so-called mini-batch gradient descent is often preferred, which lies between the extremes of batch and stochastic gradient descent since the number of randomly selected data points is greater than one but smaller than the whole training set.

The *batch size*, i.e., the number of training points we choose for mini-batch gradient descent is a further hyperparameter in the mini-batch gradient descent algorithm. Estimating the gradient from few examples might lead to a bad approximation of the gradient. However, the standard error of the mean scales with $1/\sqrt{n}$ (cf. section 8.1.3 in [14]). i.e., the quality of the gradient estimate does not even improve linearly with added computational cost. Thus, in practice, selecting small batch sizes is possible, since the degrading in the gradient approximation is usually acceptable. Typical values lie in the range [32, 256] and are powers of 2, for which GPUs run faster. Moreover, small batches can provide regularization [142]: for instance, mini-batch training can follow curves opposed to batch training which in one epoch only takes a single step and thus a straight line.

SGD methods can be improved using so-called momentum [143]. Considering a physical analogy, the negative gradient is representing a force that pushes a "particle" (the weight) through space (the loss landscape) while respecting Newton's law of motion. Assuming unit mass, the velocity corresponds to the momentum. To move towards the minimum, one employs the gradient to change the velocity rather than the position of the particle [144]. Compared to Eq. 2.2.2.1, the update rule changes to [14]

$$\mathbf{v} \leftarrow \alpha\mathbf{v} - \epsilon\nabla\hat{J}(\boldsymbol{\omega}), \tag{2.2.2.2}$$

$$\boldsymbol{\omega} \leftarrow \boldsymbol{\omega} + \mathbf{v}. \tag{2.2.2.3}$$

By introducing momentum, information from previous gradients is included, as controlled by the hyperparameters α and ϵ. This way, SGD finds directions based on a (leaky) weighted average of past gradients. To gain intuition, one can perceive the optimization process as a heavy ball rolling down a landscape, gaining momentum for steep descents and decelerating for ascents. If previous gradients have all been pointing in similar directions, larger steps will be taken and regions with high curvature will be strided faster. For instance, if the error surface is simply a tilted plane, momentum will be much faster than standard gradient descent. Information from previous gradients can also help to avoid directions where progress will be much slower.

There is a class of optimization algorithms that adapt the learning rate differently for each parameter. This is contrast to momentum, which applies uniform changes only. At the heart of these approaches is the insight that different directions in parameter space are associated with different sensitivity of the loss function. In a first simple approach (formalized in the so-called delta-bar-delta algorithm [145]), one inspects the partial derivative of the cost function with respect to each parameter and increases (decreases) the learning rate if the sign is constant (is varying). In the so-called AdaGrad algorithm [146], parameters corresponding to large (small) gradient receive significant down- (up-) scaling, resulting in the update rule [14]

$$\mathbf{r} \leftarrow \mathbf{r} + \nabla\hat{J}(\boldsymbol{\omega}) \odot \nabla\hat{J}(\boldsymbol{\omega}), \tag{2.2.2.4}$$

$$\boldsymbol{\omega} \leftarrow \boldsymbol{\omega} - \frac{\eta}{\sqrt{\delta + \mathbf{r}}} \odot \nabla \hat{J}(\boldsymbol{\omega}). \tag{2.2.2.5}$$

Here, $\odot$ denotes the element-wise product. Division and square root are applied element-wise as well. For each iteration, the gradient is calculated and its squared value accumulated in the variable $\mathbf{r}$ (cf. Eq. 2.2.2.4). The parameters are updated using an adaptive learning rate (cf. 2.2.2.5) which is scaled differently for each parameter according to the accumulated squared gradients. In Eq. 2.2.2.5, η is the learning rate and δ a constant that prohibits division by zero. The goal of this scaling is to achieve greater advancement in directions of parameter space that have a more gentle slope. While providing advantages in convex-optimization problems, the reduction of the learning rate imposed by AdaGrad can bee too restrictive in non-convex problems. This includes the optimization of deep neural networks, whose loss landscape contains rich structure through which the learning trajectory can pass before converging within a locally convex region. AdaGrad, however, may miss these convex regions as it employs the full history of gradients that may decrease the learning rate too strongly (cf. section 11.7 of [52]). In particular, the variable $\mathbf{r}$ increases in linear and unbounded fashion which translates accordingly to the learning rate.

RMSProp [144] introduces means to forget more extreme parts of the accumulated gradient history, enabling fast convergence after finding convex regions. To this end, an exponentially weighted moving average is introduced via

$$\mathbf{r} \leftarrow \rho \mathbf{r} + (1-\rho) \nabla \hat{J}(\boldsymbol{\omega}) \odot \nabla \hat{J}(\boldsymbol{\omega}), \tag{2.2.2.6}$$

$$\boldsymbol{\omega} \leftarrow \boldsymbol{\omega} - \frac{\eta}{\sqrt{\delta + \mathbf{r}}} \odot \nabla \hat{J}(\boldsymbol{\omega}). \tag{2.2.2.7}$$

ρ is the decay rate, determining the length scale of the leaky or moving average, i.e., how much of the history is considered. By iteratively applying Eq. 2.2.2.6, one finds that the accumulate of gradients converges to $1/(1-\rho)$ (cf. section 11.8 in [52]), i.e., it is normalized to one. Eq. 2.2.2.6 is similar to the leaky average used for momentum (cf. Eq. 2.2.2.2, 2.2.2.3).

Adam [10] is a robust optimization algorithm that borrows and extends characteristics of the so-far discussed methodologies. Exponential weighted moving averages are used to estimate momentum and second moment (the variance) of the gradient via

$$\mathbf{s} \leftarrow \rho_1 \mathbf{s} + (1-\rho_1) \nabla \hat{J}(\boldsymbol{\omega}), \tag{2.2.2.8}$$

$$\mathbf{r} \leftarrow \rho_2 \mathbf{r} + (1-\rho_2) \nabla \hat{J}(\boldsymbol{\omega}) \odot \nabla \hat{J}(\boldsymbol{\omega}). \tag{2.2.2.9}$$

ρ_1, ρ_2 are two non-negative hyperparameters. Initializing $\mathbf{v}, \mathbf{s}$ to the origin introduces a bias to smaller values, leading to slow learning start. This bias is corrected by the rescaling

$$\hat{\mathbf{s}} = \frac{\mathbf{s}}{1-\rho_1^t}, \ \hat{\mathbf{r}} = \frac{\mathbf{r}}{1-\rho_2^t}. \tag{2.2.2.10}$$

Here, t denotes the current iteration and $\rho_i^t := (\rho_i)^t, i = 1, 2$. While RMSProp also estimates the variance, it does not include correction factors, thus being biased in early phases of the training. Finally, the update of the parameters is performed via

$$\boldsymbol{\omega} \leftarrow \boldsymbol{\omega} - \eta \frac{\hat{\mathbf{s}}}{\sqrt{\hat{\mathbf{r}}} + \delta}. \tag{2.2.2.11}$$

Beyond Adam, further modifications are reported [147] including second order methods such as Newton's method, conjugate gradients, and the Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm [148], which is employed in materials science for local optimization of atomic structures. We refer to section 8.6 of [14] for more details.

To conclude this section, we shortly want to discuss weight initialization. Let us consider two hidden units that have the same biases as well as in- and outgoing weights. Then, their gradients will be the same and thus it is not possible to learn that they are different features [144]. Thus, symmetry must be broken, e.g., via random initialization. More advanced techniques such as described in [149] sample from a uniform distribution, while limiting the uniform distribution based on the number of input and output units.

2.2.3 Regularization

The ultimate goal is to design ML algorithms that generalize well to unseen data. This amounts to reducing the (empirical) test error. Techniques that address this central task are referred to as regularization methods. We mainly follow chapter 7 of [14] and section 3.2.2 of [124].

One way to regularize a ML model is to reduce model capacity via penalizing the parameters. Specifically, one adds a penalty term to the loss function, which yields

$$\tilde{J}(\boldsymbol{\omega}) := J(\boldsymbol{\omega}) + \lambda\Omega(\boldsymbol{\omega}). \tag{2.2.3.1}$$

A common choice of penalty is $\Omega(\boldsymbol{\omega}) = \frac{1}{2}||\boldsymbol{\omega}||_2^2$, the ℓ^2 norm of the parameters which is also called weight decay. Intuitively, this extra term pushes the weights closer to the origin. Other alternatives are ℓ^1 regularization, $||\omega||_1 = \sum_i |\omega_i|$, which leads to a preference of sparse parameter settings. This property is being used for feature selection methods, such as the so-called Least Absolute Shrinkage and Selection Operator (LASSO) [150].

Generalization ability can also be improved by augmenting the dataset. Considering the example of image classification, a typical way to expand the training set is to include differently rotated and scaled images. This way, the network can learn these invariances and generalizes better to out-of-training samples. As we discussed in section 1.4 in a materials science context, learning physical symmetries such as rotations via data augmentation is unreliable. The addition of noise to training points is a further straightforward way to increase the dataset size, for both training and test data. For instance, one may add Gaussian noise to the images. In this thesis, we distort the atomic positions or remove atoms in random fashion to generate the test set. For the training set, we employ different hyperparameter settings of the materials representation (cf. section 4.3.3.2).

Injection of noise can be applied not only to the model inputs but also the model parameters. This leads to the class of stochastic regularization techniques. A popular representative is dropout [151, 152] where neurons are randomly dropped in each layer during training (cf. Fig. 2.4). Usually, dropout is only used at training time to reduce overfitting. Specifically, the goal is to avoid over-specialization and co-adaption of individual neurons. Dropout is related to the concept of ensemble methods, where the predictions of a set of possibly weak and separately trained learners are combined (for instance, via averaging) to improve the generalization error. For so-called bagging methods, n models are trained on n different datasets (sampled randomly with replacement). At test time, each participant of the ensemble would have to be evaluated, which can be impractical for deep neural networks. Dropout approximates bagging via the following modification of mini-batch stochastic gradient descent. For each point i in each batch, a binary mask vector $\mathbf{m}_i^l$ is sampled for all user-defined layers l of the network (except the output layer). The components of $\mathbf{m}_i^l$ are sampled independently and are set to zero with a certain probability p_i, a hyperparameter that is called dropout ratio. The same masks are used for forward and backpropagation. In contrast to bagging, dropout produces models that are not independent and share parameters that are random realizations of a predefined parent architecture. This way, exponentially many models can be represented with reasonable memory requirements. Similar to bagging, the dropout subnetworks are trained on subsets, the batches, corresponding to random samples with replacement. However, most of the subnetworks are either not trained at all – since they are missed by the random sampling – or only for a single step. Still, due to parameter sharing, acceptable parameter settings can be reached for all subnetworks. To obtain a prediction, bagging involves the combination of all submodel predictions, e.g., via averaging. For dropout, this would involve an average of exponentially many submodels. A practical approximation is the scaling of each unit i by $1/1-p_i$ which accounts for the missing neurons at training time. This way, one tries to approximately match the expected inputs to unit i at test and training time. Dropout can also be applied at test time, which is referred to as Monte Carlo (MC) dropout, allowing a Bayesian interpretation of the neural network model. We will discuss this in more detail in section 2.3.2. Note that instead of sampling the dropout masks from a Bernoulli distribution, one may also employ other probability distributions, e.g., Gaussian multiplicative noise [152]. Using multiplicative noise enforces a type of robustness that would not be guaranteed by using, for instance, additive noise. Specifically, the neurons could adopt very large values during optimization such that the added noise becomes negligible.

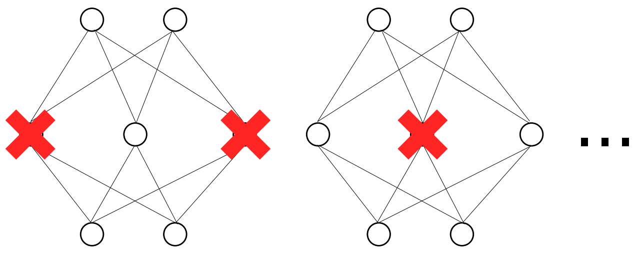
Figure 2.4: Illustration of dropout regularization.

Finally, we want to discuss *early stopping*. A typical scenario is that at the beginning of training, the validation error decreases simultaneously with the training error but then starts to rise again, while the training error continues to decline. Thus, model parameters from early training epochs might actually outperform those extracted from the end of training. In early stopping, the model parameters are saved each time the validation error decreases. At the end of training, the optimal set of parameters is returned. This simple strategy may be extended by selecting a desired amount of increase in model performance for which model parameters will be saved. One may even abort the training process if no improvement is made for a predefined number of epochs. Intuitively, early stopping can be perceived as restricting the volume of parameter space [153, 154] and is equivalent to weight decay for linear models with quadratic cost function [14]. Section 4.3.3.3 discusses how we employ early stopping in this thesis.

2.2.4 Hyperparameter tuning via Bayesian optimization

A reoccurring scenario in ML is the optimal selection of hyperparameters. In particular, these parameters are not fixed by the learning algorithm but can have significant influence on model performance (both training and generalization error). In the previous sections, we encountered multiple hyperparameters: architecture specifics (number of layers, number of neurons), regularization parameters (weight decay, dropout ratio), or optimization-related settings (learning rate and associated parameters for its adaption during training). After a specific hyperparameter selection, training is performed and the quality of the choice is evaluated using the error on the validation set. If the model performance is not satisfying, new hyperparameters may be selected. This hand-tuning approach to hyperparameter optimization is heavily based on human intuition, hindering understandability and reproducibility of research results. In particular, it is often hard to distinguish approaches that are fundamentally improving or just involve more fine-tuning. Establishing algorithms for automatic hyperparameter selection is a significant step towards addressing this important problem [155].

A first step towards automatic hyperparameter search is to define a specific range of values for each hyperparameter and compute the validation error for all possible combinations. This is also referred to as grid search. Alternatively, one may randomly sample from the predefined hyperparameter intervals, so-called random search [156]. This strategy often outperforms grid search, in particular if the search space is large and testing every combination is computationally infeasible. Both grid and random search are prone to the probing of uninteresting corners of search space, since no memory of previous samples or an estimate of the expected reward is included. In particular in deep learning, cost functions are computationally expensive due to large datasets and model complexity. Bayesian optimization is an attempt to address these issues.

Bayesian optimization is a general approach for the global optimization of complex and computationally expensive, black-box like functions f (see also [157–159] for more details). In hyperparameter optimization, f may correspond to scores that depend on configurations of hyperparameters, e.g., scalar performance measures such as the classification accuracy defined with respect to a validation set and a specific model architecture. A central task is to model the distribution of scores y given configurations x, i.e., $p(y|x)$. This conditional distribution is determined by updating a prior from a history of previously explored configurations and associated scores, denoted as $\mathcal{H} = \{(x, f(x)\}$. The model for $p(y|x)$ should be computationally tractable to enable rapid evaluation. Gaussian processes [160] or decision trees [161] may be used to model $p(y|x)$ which is updated iteratively in a Bayesian fashion. Since $\mathcal{H}$ changes the model, we denote the approximation of $p(y|x)$ as $p_M(y|x, \mathcal{H})$. This strategy relies on the evaluation and proposal of new candidate configurations, for which expected improvement (EI) is a typical measure [162] that is defined as [155]

$$\mathrm{EI}_{y^*}(x) = \int_{y<y^*} y \; p_M(y|x, \mathcal{H}) dy. \tag{2.2.4.1}$$

This expression approximates the expectation (according to the model $p_M(y|x, \mathcal{H})$) that the original, to be optimized function is below a certain threshold y^* for a specific configuration x. The established terms are employed in so-called sequential model-based global optimization algorithms [163]: The computationally costly function f is optimized for a predefined number of iterations T by optimizing a surrogate model of f. Specifically, a new point x^* is suggested that optimizes the surrogate model, e.g., by maximizing the expected improvement. Then, the function f is evaluated – only once per iteration – for the point x^*, yielding a new pair $(x^*, f(x^*))$ that is added to the history $\mathcal{H}$. This updated history is then used to fit a new model, after which a new iteration starts.

The Tree-structured Parzen estimator (TPE) [155,163] is a specific hyperparameter optimization algorithm. It assumes a tree structure in hyperparameter space where leaf variables (such as the number of neurons in a layer) depend on the definition of the node variables (such as the number of neural-network layers). After defining prior distributions for each hyperparameter, TPE relies on the fitting of Gaussian mixture models (GMMs). In general, GMMs estimate the density underlying a given dataset as a superposition of Gaussian densities, which are defined by mean and covariance matrix (see section 6.8 in [33]). TPE is implemented in the Python package hyperopt [164] which provides a general automatic framework for hyperparameter optimization.

2.3 Bayesian deep learning

So far, we considered ML models only as learning systems that provide a prediction. In many situations, it is useful to quantify the confidence associated with a particular prediction, i.e., how much one can trust the model (see also section 1 of [124]). For instance, let us consider an image classifier that is trained on dogs and cats. When faced with a completely different image, for instance, a jaguar, the model may predict "cat“. In this case, it would be desirable to have an additional indicator signaling that the model is not confident, i.e., one needs to quantify the model uncertainty. One can distinguish two sources of uncertainty [124,165,166]: So-called epistemic uncertainty (stemming from the Greek word “episteme” that is translated as “knowledge”) corresponds to the lack of knowledge due to the model parameters being insufficient to describe the data. Adding more data can reduce this kind of noise. Moreover, so-called aleatoric uncertainty (from the Latin word “aleator” meaning “dice player”) is caused by noise in the observations, e.g., due to experimental inaccuracies. Simply expanding the dataset will not eliminate this uncertainty source.

Bayesian modeling can be employed to quantify the uncertainty of a ML model [158]. Applied to deep neural networks, this leads to the field of Bayesian deep learning. Standard neural networks, as discussed in section 2.1, are unable to provide reliable model classification uncertainty [123]: Considering a classification setting, there is widespread use of the probability provided by the last layer as uncertainty estimate. These probabilities are typically obtained by normalizing the sum of output values using the so-called softmax activation function (cf. Eq. 2.1.2.8). The class with maximal probability corresponds to the final prediction (in this thesis, of a specific structural class). One may interpret the classification probability as quantification of model confidence. For instance, given a model that is trained to distinguish dogs and cats, one may compare two predictions that differ in classification probability and interpret the prediction with higher classification probability as more reliable. However, this strategy is unreliable as standard neural networks tend to erroneously assign unjustified high confidence to points for which a low confidence should be

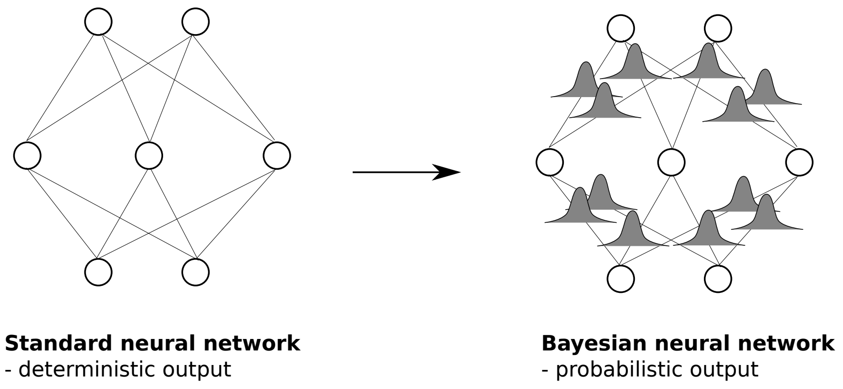

Figure 2.5: Illustration of the difference between parameters and output of standard (left) and Bayesian neural networks (right). While in standard neural networks, parameters are fixed and produce deterministic output, in Bayesian neural networks, distributions are placed over the model parameters, resulting in probabilistic outputs.

returned instead. In particular, even if a model has high softmax output for a given input, the predictive uncertainty can still be high [123]. The main reason for this behavior is that standard-neural-network predictions are deterministic, with the softmax output providing only point estimates of the true probability distribution of outputs. In Bayesian neural networks, this is addressed by placing distributions over model parameters (cf. Fig. 2.5). This results in probabilistic outputs – in contrast to the point estimates from deterministic neural networks– from which uncertainty estimates can be obtained. Gal and Ghahramani [123] showed that high-quality uncertainty estimates (alongside predictions) can be calculated at low cost using stochastic regularization techniques such as dropout [151,152]. In particular, the uncertainty estimates are principled in the sense that they approximate those of a well-known probabilistic model, the Gaussian process [160]. We will discuss this in section 2.3.2 in more detail.

2.3.1 Bayesian approach and variational inference

For the formalization of model uncertainty quantification, Bayesian modeling is employed. We mainly follow section 2 of [124]. In a frequentist approach, the probability of a certain event corresponds to the relative frequency of the event as determined from repeated experiments. Bayesian probability allows to incorporate prior knowledge or belief which is encoded in the *prior distribution.* For instance, the assumption of a fair coin, i.e., $p(\text{heads}) = p(\text{tails}) = 0.5$, is a reasonable prior. Specializing to a ML problem, we are given inputs $\mathbf{X}$ and outputs $\mathbf{Y}$. The goal is to find the most likely parameters $\boldsymbol{\omega}$ of the model $f_{\boldsymbol{\omega}}(\mathbf{x})$ that generated the outputs. To apply Bayesian modeling, one first has to define a prior distribution $p(\boldsymbol{\omega})$ over the model parameters, expressing the initial belief in the more likely parameters before any data point is observed. With incoming observations, the prior is updated via Bayes' theorem to obtain the *posterior distribution*, which captures the most likely model parameters given the data. Specifically, the posterior distribution is determined via

$$p(\boldsymbol{\omega}|\mathbf{X},\mathbf{Y}) = \frac{p(\mathbf{Y}|\mathbf{X},\boldsymbol{\omega})p(\boldsymbol{\omega})}{p(\mathbf{Y}|\mathbf{X})}. \tag{2.3.1.1}$$

$p(\mathbf{Y}|\mathbf{X},\omega)$ denotes the *likelihood distribution*, which in a classification task can be obtained via the softmax likelihood

$$p(y=c|\mathbf{x},\boldsymbol{\omega}) = \frac{\exp\left([f_{\boldsymbol{\omega}}(\mathbf{x})]_c\right)}{\sum_{c'}\exp\left([f_{\boldsymbol{\omega}}(\mathbf{x})]_{c'}\right)}. \tag{2.3.1.2}$$

Furthermore, $p(\mathbf{Y}|\mathbf{X})$ is called *model evidence* and can be calculated via integrating (also called marginalizing) $p(\mathbf{Y}|\mathbf{X},\boldsymbol{\omega})$ over $\boldsymbol{\omega}$, i.e.,

$$p(\mathbf{Y}|\mathbf{X}) = \int p(\mathbf{Y}|\mathbf{X},\boldsymbol{\omega})p(\boldsymbol{\omega})d\boldsymbol{\omega}. \tag{2.3.1.3}$$

Given the posterior, one can perform inference, i.e., calculate the prediction $\mathbf{y}'$ given a new input $\mathbf{x}'$ via

$$p(\mathbf{y}'|\mathbf{x}',\mathbf{X},\mathbf{Y}) = \int p(\mathbf{y}'|\mathbf{x}',\boldsymbol{\omega})p(\boldsymbol{\omega}|\mathbf{X},\mathbf{Y})d\boldsymbol{\omega}. \tag{2.3.1.4}$$

To compute the posterior, integrals over the model parameters $\boldsymbol{\omega}$ need to be calculated. Only for simple models the integration over all possible $\boldsymbol{\omega}$ is feasible, while typically approximation techniques are required – especially for deep-neural networks.

Variational inference is a general method to approximate intractable probability distributions as they frequently appear in Bayesian inference. Specifically, the model posterior is approximated with a distribution $q_\theta(\boldsymbol{\omega})$, parametrized by θ. In particular, $q_\theta(\boldsymbol{\omega})$ should be less complex and easier to evaluate than the model posterior. To determine $q_\theta(\boldsymbol{\omega})$, one can minimize the *Kullback–Leibler* (KL) divergence which is defined as

$$\mathrm{KL}(q_\theta(\boldsymbol{\omega}) \mid\mid p(\boldsymbol{\omega}|\mathbf{X},\mathbf{Y})) = \int q_\theta(\boldsymbol{\omega}) \log \frac{q_\theta(\boldsymbol{\omega})}{p(\boldsymbol{\omega}|\mathbf{X},\mathbf{Y})} d\boldsymbol{\omega}. \tag{2.3.1.5}$$

This way, marginalization (the calculation of integrals) is replaced with optimization (the calculation of derivatives), which is usually more tractable. We refer to [167] and [124] for more details. The minimum of the KL divergence, denoted as $q'_\theta(\boldsymbol{\omega})$, is then used to approximate the inference process in Eq. 2.3.1.4 via

$$p(\mathbf{y}'|\mathbf{x}',\mathbf{X},\mathbf{Y}) \approx \int p(\mathbf{y}'|\mathbf{x}',\boldsymbol{\omega})q'_\theta(\boldsymbol{\omega})d\boldsymbol{\omega}. \tag{2.3.1.6}$$

While replacing the intractable posterior, this equation still contains expensive calculations, in particular the integral over parameter space and the evaluation of the integrand for the whole dataset. This limitation is addressed by Monte Carlo dropout that is discussed in the next section. Specifically, one can show that variational inference and neural-network optimization using dropout regularization are equivalent. In particular, a neural network trained with dropout can be interpreted as a Bayesian neural network.

2.3.2 Monte Carlo Dropout

As discussed at the beginning of this chapter, one can think of Bayesian neural networks as standard neural networks with distributions being placed over the model parameters. This results in probabilistic outputs from which uncertainty estimates can be obtained. The major drawback is that training and inference via traditional Bayesian neural networks is in general difficult, since it requires to solve high-dimensional and thus computationally costly integrals. For classification, expensive calculations are required to determine $p(y = c|\mathbf{x}, D_{\text{train}})$, which is the probability that the classification is assigned to a class c, given input $\mathbf{x}$ and training data D_{train}. Then, for a specific input $\mathbf{x}$ (in our case the SOAP descriptor), the most likely class c, i.e., the one with largest $p(y = c|\mathbf{x}, D_{\text{train}})$ is the predicted class.

Gal and Ghahramani [123] showed that stochastic regularization techniques such as dropout [151,152] can be used to calculate high-quality uncertainty estimates (alongside predictions) at low cost. This approach is also called Monte Carlo (MC) dropout. In dropout, neurons are randomly dropped in each layer before the network is evaluated for a given input (cf. Fig. 2.4). Usually, dropout is only used at training time with the goal of avoiding overfitting by preventing over-specialization of individual units. Keeping regularization also at test time allows to quantify the uncertainty. Practically, given a new input, one collects and subsequently aggregates the predictions while using dropout at prediction time. This gives a collection of probabilities being denoted as $p(y = c|\mathbf{x}, \boldsymbol{\omega}_t)$, which is the probability of predicting class c given the input $\mathbf{x}$ at a specific forward-pass t, with model parameters $\boldsymbol{\omega}_t \sim q'_\theta(\boldsymbol{\omega})$. From this collections of probabilities, one can estimate the actual quantity of interest, $p(y = c|\mathbf{x}, D_{\text{train}})$, by a simple average [123]

$$p(y = c|\mathbf{x}, D_{\text{train}}) \approx \frac{1}{T} \sum_{t=1}^{T} p(y = c|\mathbf{x}, \boldsymbol{\omega}_t), \tag{2.3.2.1}$$

where T is the number of forward-passes. Typically, values of $10^2 - 10^3$ suffice to obtain good results (see also section 4.3.3.1 for more details on how we choose this parameter).

While the average can be used to infer the class label c, additional statistical information, which reflects the predictive uncertainty, is contained in the collected forward-passes, i.e., the probabilities $p(y = c|\mathbf{x}, \boldsymbol{\omega}_t)$. These effectively yield a histogram for each class and define, when varying over all possible c, a (discrete) probability distribution. For classification, several uncertainty quantifiers are reported [123,165]. For instance, one may employ *mutual information*, which is well-grounded in information theory.

In our case, for a given test point x, the mutual information between the predictions and the model posterior $p(\boldsymbol{\omega}|D_{\text{train}})$ is defined as (where we adapt the notation of [123,124,168])

$$\mathbb{I}\left[y, \boldsymbol{\omega}|\mathbf{x}, D_{\text{train}}\right] := \mathbb{H}[y|\mathbf{x}, D_{\text{train}}] - \mathbb{E}_{p(\boldsymbol{\omega}|D_{\text{train}})}\left[\mathbb{H}[y|\mathbf{x}, \boldsymbol{\omega}]\right]. \tag{2.3.2.2}$$

The first term on the right-hand side is the *predictive entropy* [124] that quantifies the (average) information in the distribution of predictions and is given by

$$\mathbb{H}[y|\mathbf{x}, D_{\text{train}}] := -\sum_{c} p(y = c|\mathbf{x}, D_{\text{train}}) \log p(y = c|\mathbf{x}, D_{\text{train}}). \tag{2.3.2.3}$$

Both mutual information and predictive entropy are strictly greater than zero. Thus, the mutual information is bounded from above by the predictive entropy [165]. The second term in 2.3.2.2 is defined as

$$\mathbb{E}_{p(\boldsymbol{\omega}|D_{\text{train}})}\left[\mathbb{H}[y|\mathbf{x},\boldsymbol{\omega}]\right] := \mathbb{E}_{p(\boldsymbol{\omega}|D_{\text{train}})}\left[\sum_c p(y=c|\mathbf{x},\boldsymbol{\omega})\log p(y=c|\mathbf{x},\boldsymbol{\omega})\right]. \tag{2.3.2.4}$$

which one may call expected entropy as it averages the entropy of the predictions given the parameters $\boldsymbol{\omega}$ that are distributed according to the posterior distribution [165]. Using MC dropout, one can approximate the mutual information via [123]

$$\begin{aligned}&\mathbb{I}\left[y,\boldsymbol{\omega}|\mathbf{x},D_{\text{train}}\right] \approx \\ &-\sum_c\left(\frac{1}{T}\sum_t p\left(y=c|\mathbf{x},\boldsymbol{\omega}_t\right)\right)\log\left(\frac{1}{T}\sum_t p\left(y=c|\mathbf{x},\boldsymbol{\omega}_t\right)\right) \\ &+\frac{1}{T}\sum_c\sum_t p\left(y=c|\mathbf{x},\boldsymbol{\omega}_t\right)\log p\left(y=c|\mathbf{x},\boldsymbol{\omega}_t\right).\end{aligned} \tag{2.3.2.5}$$

Note that the predictive entropy can also used to quantify the uncertainty. There are also other measures (cf., for instance, [165]) while mutual information and predictive entropy work well in practice.

To gain intuition on mutual information and predictive entropy, let us consider an example from crystal classification (inspired by section 3.3 in [124]). The problem setting is to distinguish two crystal structures, say fcc and bcc, which are represented as model outputs $(1,0)$ and $(0,1)$, respectively. There are now two extreme cases:

1. The collected probability vectors are $\{(1,0),(1,0),(1,0),..\}$, i.e., they all coincide and predict the same class label with $100\,\%$ probability.

2. The collected probability vectors are $\{(1,0),(0,1),(1,0),(0,1),...\}$, i.e., they alternate between the two classes.

In these scenarios, mutual information and predictive entropy coincide: For 1., they are zero, i.e., the model is highly confident, while for 2. they are maximal (which equals the logarithm of the number of classes), i.e., the model is highly uncertain. A third scenario is as follows:

3. The collected probability vectors are $\{(0.5,0.5),(0.5,0.5),(0.5,0.5),(0.5,0.5),...\}$, i.e., they all coincide but each class is assigned a $50\,\%$ probability.

In this case, the mutual information will vanish but the predictive entropy will be maximal. In particular, the mutual information will vanish any time the collected probabilities are constant along the different MC-dropout steps (e.g., also if the probabilities are (0.7, 0.3)). Thus, following [124], mutual information quantifies *model confidence.*

Chapter 3

Unsupervised learning

In unsupervised learning, the goal is to learn hidden patterns and concepts from unlabeled data. So far, we discussed supervised prediction of targets $\mathbf{Y}$ from inputs $\mathbf{X}$, i.e., we investigated properties of $P(\mathbf{Y}|\mathbf{X})$. In unsupervised learning, the goal is to infer the properties of $P(\mathbf{X})$, the joint density of observations, without supervision (or rewards from the environment in a reinforcement learning setting). We refer to section 14 of [33] and [169] for broader overviews on unsupervised-learning techniques.

In the following sections, we discuss two unsupervised-learning methods, which we use to analyze and explain the predictions of our neural-network approach ARISE: clustering (section 3.1) and dimensionality reduction (section 3.2). Specifically, we employ the *Hierarchical Density-based Spatial Clustering Applications with Noise* (*HDBSCAN*) framework (section 3.1.3), which is based on *Density-based Spatial Clustering Applications with Noise* (*DBSCAN*) that is discussed in section 3.1.1 and 3.1.2. For dimensionality reduction, we first discuss some basic concepts and then explain a particular type of manifold learning algorithm in section 3.2.1, *Uniform Manifold Approximation and Projection* (*UMAP*).

3.1 Clustering

The goal of clustering methods is to separate a given dataset into groups, the clusters. Points within a cluster are similar to each other and distinct from all other points. The definition of similarity within and between clusters allows to differentiate clustering methods.

More formally, a minimal definition of clustering requires the set of data points $\mathbf{x} \in O$ (typically $O \subset \mathbb{R}^d$) and a distance function d mapping pairs of points to a positive real number, i.e., $d : O \times O \to \mathbb{R}^+$ [170, 171].

3.1.1 Density-based clustering

In many approaches, a dataset with m points is partitioned into a predefined number of k groups. Using the distance function d, the similarity of data points belonging to the same cluster is maximized while the similarity to points outside the cluster is minimized. These methods can be understood as parametric, i.e., the data density $P(\mathbf{X})$ is considered as a mixture of k densities $p_i(\mathbf{X}), i = 1, ..., k$. These densities are members of a specific parametric family such as Gaussian distributions (e.g., GMMs). The unknown parameters of the k densities are estimated from the data. The discovered clusters are typically convex-shaped.

Density-based clustering defines clusters as contiguous high-density regions that are detached from each other via contiguous low-density regions. This approach is non-parametric and does not require the user to fix the number of clusters. Moreover, the clusters are not restricted to convex shape.

To illustrate the concept of density-based clustering, let us first consider a simplified example in Fig. 3.1. From an estimate for the density underlying the data, one can select a threshold that results in different cluster assignments. In this example, a parametric approach may lead to comparable results. To demonstrate the advantage of density-based clustering methods, we consider a more complicated example in Fig. 3.2a: One can see that the shape of the black cluster is spherical to a good approximation and thus may be captured via parametric methods such as GMMs. The red cluster, however, is non-spherical and all of its points cannot be captured: for instance, methods such as k-means [172] can only detect spherical regions and thus will detect only part of the red cluster. In general, detecting arbitrarily shaped clusters is not possible with parametric methods. The choice of parametric family limits the shape of the detected clusters. Furthermore, the number of clusters is a hyperparameter that has to be chosen carefully. Fig. 3.2 also contains points that are more distant to the rest of the points, and thus would be candidates for outliers (a differentiation that cannot be achieved using, for instance, k-means). One may recover some of the outliers by deciding a lower bound for the density, i.e., all points below a certain density are considered outliers (e.g., for GMMs, below a multiple of the standard deviation σ). Conversely, a density-based approach is non-parametric (no parametric family is chosen) and thus imposes less strict assumptions on the data: Number and shape of clusters is arbitrary and solely inferred from the data. Furthermore, there is the option to refuse the assignment of a given data point to a cluster, i.e., to identify "outliers". Note that this term requires careful definition. The methods employed in this thesis follow the definition by Hawkins [173], which considers an outlier *"an observation that deviates so much*

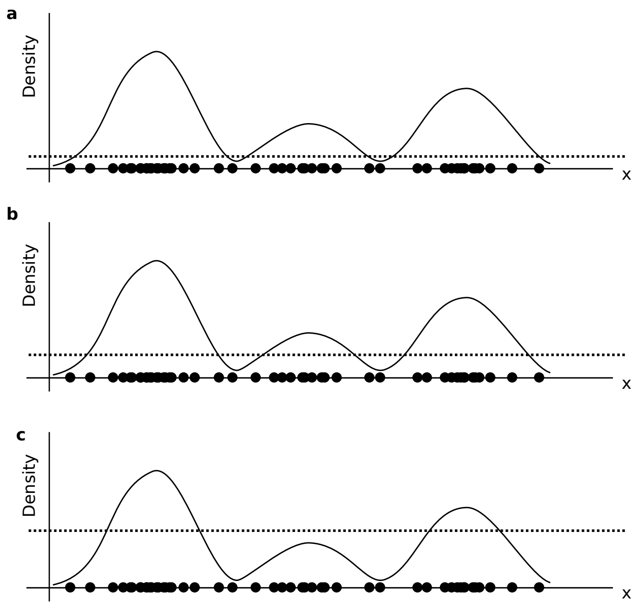

Figure 3.1: Illustration of density-based clustering using an artificial 1D dataset (inspired by Fig. 1 in [171]). A specific cut of the estimated density (y-axis) results in a certain number of connected components. In each of these, the data points exceed the density threshold. All points below the density level are considered noise. These assignments depend on the chosen cut being either very low (**a**), intermediate (**b**) or very high (**c**). In particular, in case **c**, one cluster is missed.

from other observations as to arouse suspicion that it was generated by a different mechanism." Coming back to Fig. 3.2a, a non-parametric method is able to separate all clusters and even refuse to cluster points that do not fall into a regular pattern (such as the gray points). Furthermore, one can observe that the (Euclidean) distance between points in the same cluster can be larger as the distance between points that belong to different clusters. Still, the density is high in the cluster regions. In summary, one can see that in comparison to parametric approaches, non-parametric methods allow the data more to speak for itself. That is why these kinds of techniques are particularly attractive for performing exploratory data analysis – one major point in this thesis (e.g., section 6.6.2). Different density-based approaches can be discriminated by the way in which the density is estimated. A concrete example is considered in the next section.

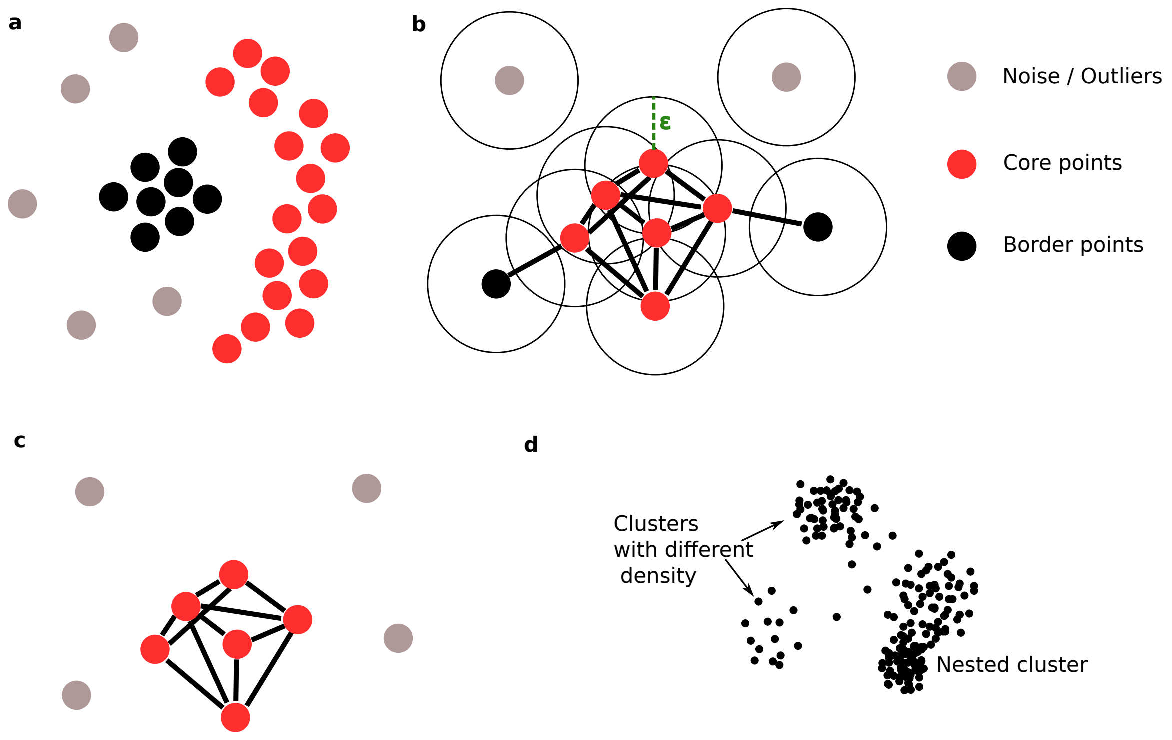

Figure 3.2: Artificial datasets demonstrating the concept behind DBSCAN (and its variant DBSCAN* [174]). **a** 2D dataset in which both spherical (black points) and non-convex clusters (red points) appear, alongside points being scattered across the map (in gray). **b** Illustration of the connectivity that arises within the DBSCAN algorithm from a particular choice of radius ϵ (highlighted in dark green) and the definition of *core*, *border*, and *noise points* (or rather *outliers*). While the noise points can be clearly distinguished, the difference between core and border points is due to the predefined density threshold k, corresponding to the number of points within ϵ. **c** Graph representation of the connectivity. Compared to **b**, the definitions of the DBSCAN* algorithm are employed in which all non-core points are considered outliers. **d** 2D dataset that demonstrates the occurrence of different types of clusters within a single dataset, in particular clusters with different density (left) or nested and sub-clusters (right). This example demonstrates the limitations of DBSCAN and its variant DSBCAN* [174] (see also the main text in section 3.1.2).

3.1.2 DBSCAN

DBSCAN [175] is an example of non-parametric density-based clustering. In this approach, the density is approximated by defining the connectivity between points using a distance metric (e.g., the Euclidean distance). Given a set of points $\mathbf{X}$, two thresholds are chosen: ϵ for the pairwise distance and k for the density (which is the minimum number of points with distance $\leq \epsilon$). Then, the following data points are distinguished [174,176] (see also Fig. 3.2b for an illustration of the following concepts):

- $\mathbf{x}_i \in \mathbf{X}$ is called a *core point* if its number of neighbors within ϵ distance exceeds k.
- $\mathbf{x}_i \in \mathbf{X}$ is called directly *density-* or *ϵ-reachable* from a core point $\mathbf{x}_j \in \mathbf{X}$ if $\mathbf{x}_i$ is within ϵ distance from $\mathbf{x}_j$.
- $\mathbf{x}_i$ and $\mathbf{x}_j$ are called *density-connected* if they are directly or transitively ϵ-reachable, where the latter notion implies that there is a path of points which are all core points.
- All points which are not reachable from any other point are called *outliers* or *noise points*.

Using these definitions, a cluster is defined as a set of points for which every pair of objects is density-connected. Points that do not posses the core property may be part of a cluster as well and are called *border points*. The clusters are then identified as the connected components of a graph G, in which the vertices correspond to the data points $\mathbf{X}$ and edges between two vertices are drawn if and only if both are ϵ-reachable and core points (cf. Fig. 3.2c). In a modified version of DBSCAN, so-called DBSCAN* [174], all non-core points are considered outliers (i.e., also border points). The change in connectivity is illustrated in Fig. 3.2c.

While DBSCAN offers several advantages compared to parametric methods, its limitations are clear: The threshold parameters ϵ and k have to be chosen. At worst, this requires preknowledge about the scale of the dataset – otherwise no satisfying results may be obtained. Furthermore, if a dataset contains clusters with large variations in density (and maybe also nested clusters), it may be hard to find optimal values for both ϵ and d (i.e., a trade-off between distance and density) such that all or most of the clusters can be detected. This is illustrated in Fig. 3.2d. The clustering algorithm discussed in the next section addresses these limitations.

3.1.3 HDBSCAN

HDBSCAN [177, 178] is an extension of the DBSCAN algorithm. Several limitations of DBSCAN are addressed in HDBSCAN. The dependency on distance and density thresholds is eliminated and only one, more intuitive parameter needs to be chosen. Moreover, clusters with varying density can be detected.

HDBSCAN is an *agglomerative hierarchical clustering* method. To illustrate this concept, let us consider the example shown in Fig. 3.3a. At the beginning of the algorithm, each point constitutes a separate cluster. Then, the most similar points are joined iteratively, where in this case, the similarity is defined via Euclidean distance. The algorithm stops as

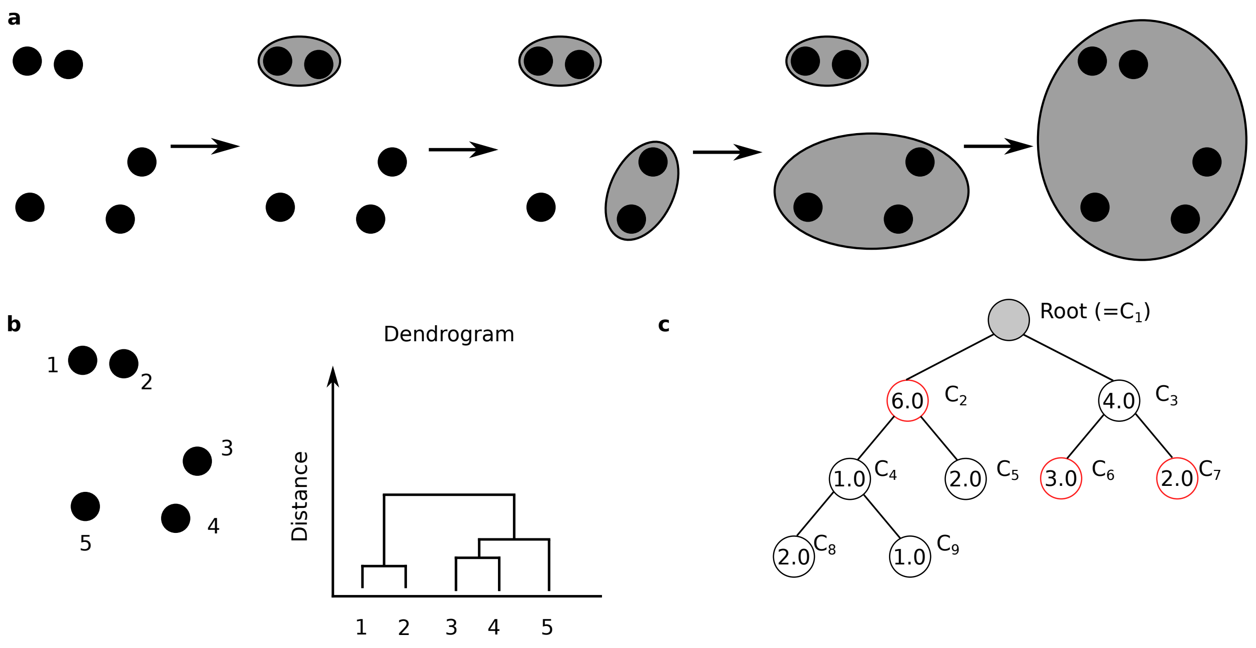

Figure 3.3: **a** Illustration of *single linkage clustering* (a specific type of *agglomerative hierarchical clustering*). **b** Construction of a *dendrogram* for the clustering procedure illustrated in **a**. **c** Illustration of the stability criterion in HDBSCAN and its consequence on the clustering assignment. This subfigure is a modified version of Fig. 2 from [174].

soon as one cluster containing all points is obtained. This procedure can be summarized in a *dendrogram*, an example of which is shown in Fig. 3.3b. In this depiction, each point in the dataset is shown on the x-axis and the y-axis (the height) corresponds to the distance. Starting from the bottom, each point constitutes a separate cluster. Going up (i.e., to larger distances), first point 1 and 2 are joined , then 3 and 4 etc. Drawing a horizontal line yields intersections with the dendrogram leafs that correspond to a specific *flat* clustering. In general, one can choose different cuts and thus obtains a hierarchy of clusters, while there is no immediate criterion which flat clustering should be preferred. Furthermore, one may try different distance metrics and linkage criteria, i.e., the condition according to which points are merged. In Fig. 3.3a two clusters are merged if they contain two points with minimal Euclidean distance. We refer to [179, 180] for additional details on hierarchical clustering.

Equipped with knowledge about density-based and agglomerative hierarchical clustering, we can now discuss the actual HDBSCAN algorithm, where we mainly follow [174, 177] and the online documentation (https://hdbscan.readthedocs.io/en/latest/how_hdbscan_works.html, version 0.8.26):

- First, a specific distance metric is employed, the *mutual reachability distance*: the *core distance* $d_{\text{core}}(\mathbf{x}_i)$ with $\mathbf{x}_i \in \mathbf{X}$ is introduced, which is the distance of $\mathbf{x}_i$ to its kth-nearest neighbor. Then, the mutual reachability distance is defined as [174]

$$d_{\text{mreach}}(\mathbf{x}_i, \mathbf{x}_j) = \max\{d_{\text{core}}(\mathbf{x}_i), d_{\text{core}}(\mathbf{x}_j), d(\mathbf{x}_i, \mathbf{x}_j)\}, \qquad (3.1.3.1)$$

where $d(\cdot, \cdot)$ denotes a metric distance (e.g., the Euclidean distance). For $\mathbf{x}_i = \mathbf{x}_j$, one defines $d_{\text{mreach}}(\mathbf{x}_i, \mathbf{x}_j) = 0$. Using mutual reachability, two dense points $\mathbf{x}_i, \mathbf{x}_j$

with a small core distance are assigned the same distance (according to $d(\cdot,\cdot)$), while sparse points will be separated by at least their core distances. This transforms the original data space, separating high-density areas from noise. There is additional, more theoretical motivation for the application of mutual-reachability distances, see, for instance, [181].

- Using Eq. 3.1.3.1, one can build a graph in analogy to the DBSCAN procedure, where again vertices correspond to data points while edges between vertices are now assigned the mutual-reachability distance. For a fully connected graph, the number of edges scales cubically with the number of points. For large datasets, this leads to large computation times. Instead of a fully connected graph, a *minimum spanning tree* is constructed. In short, this type of graph is constructed via identifying a set of edges that is minimal in the sense that removing any edge from this set would result in disconnected components.

- From the minimum spanning tree one can build a cluster hierarchy, i.e., perform the single-linkage agglomerative hierarchical clustering procedure sketched in Fig. 3.3a, while using the mutual reachability distance instead of the Euclidean distance. The hierarchy yields connected components and noise for different distance levels, containing all partitions that can be obtained via varying ϵ, d in the DBSCAN algorithm. Practically, one sorts the edges by increasing distance and then iteratively merges them into clusters.

- To extract a flat clustering, one may select a specific density threshold (corresponding to a horizontal cut through the dendrogram) and recover a DBSCAN-like procedure. However, this requires careful threshold-tuning. In particular, a choice of distance corresponds to a fixed density, leading back to the problems sketched in Fig. 3.2d. To address this, the dendrogram is transformed and simplified the following way: Going through the hierarchy from top (the largest distance) to bottom (the smallest distance), one interprets this not as one cluster splitting into ≥ 2 components but rather as a single persistent cluster from which points drop out. Traversing the hierarchy, one then examines at each split the number of points in the new clusters. If there are less than a certain number of points (denoted as the *minimum cluster size* m_{clSize}) in any of these connected components, they are not considered a "true" split but deemed noise. The distance at which these split-offs happen are saved to memory. This procedure significantly reduces the hierarchy complexity, corresponding to less dense dendrograms. To determine the flat clustering, one identifies those clusters that have the longest persistence, i.e., they "survive" longer when decreasing the distance threshold. To this end, one has to quantify the stability $S(C)$ of a cluster C. Following [174], we introduce $\lambda = \frac{1}{\epsilon}$, the inverse of the core distance. This change in notation is motivated by the fact that density-based algorithms aim to find high-density areas but so far we related to distances only. For a given point, the inverse of its kth nearest-neighbor distance (the core distance) is an efficient estimate of the local density [177], which is why it is used in the following. Increasing λ (corresponding to decreasing ϵ) traverses the hierarchy from top to bottom, reducing and splitting up clusters.

Given the definition of λ, one introduces for each cluster C the value $\lambda_{start}(C)$ as the level at which C has been separated and became a distinct cluster. Moreover, $\lambda_{end}(C)$ corresponds to the level at which C is separated into sub-clusters. Then for a point $\mathbf{x}_i$, $\lambda(\mathbf{x}_i, C)$ denotes the level beyond which $\mathbf{x}_i$ is no longer member of C, while $\lambda_{start} \leq \lambda(\mathbf{x}_i, C) \leq \lambda_{end}$. The *stability* of the cluster C is then defined as

$$S(C) = \sum_{\mathbf{x}_i \in C} \left(\lambda(\mathbf{x}_i, C) - \lambda_{start}(C)\right). \tag{3.1.3.2}$$

This stability quantifier, defined for each cluster C, can be used to extract the most persistent clusters. Specifically, one traverses the hierarchy, comparing the sum of stabilities of child and parent clusters. Considering the example shown in Fig. 3.3c, one starts from a collection of clusters (here, $C_1, ..., C_9$) and their associated cluster hierarchy tree in which each node corresponds to a cluster C_i and its stability $S(C_i)$. Starting from the lowest level, at the node C_4, the sum of stabilities of its child clusters C_8, C_9 is larger than $S(C_4)$ which is why C_4 is eliminated. Summing the stabilities of C_8, C_9, C_5 yields 5.0, which is smaller than $S(C_2)$ and thus only C_2 survives. On the other branch, the combined stability of C_6, C_7 is larger than $S(C_3)$, which is thus removed. Hence, for this artificial example, only C_2, C_6, and C_7 are selected as final clusters (highlighted in red in Fig. 3.3c). This procedure, which we sketched only exemplarily in Fig. 3.3c, can be formalized into an optimization algorithm, whose exact definition can be found in section 5 of [174].

Instead of the assignment of clustering labels (*flat clustering*), HDBSCAN can also be employed to perform *soft clustering*, where points can be part of several clusters. Specifically, points $\mathbf{x}_i$ are assigned probability vectors $\mathbf{p}$ whose jth correspond to the probability that the point is part of cluster j. In the previous paragraph, we introduced the point- and cluster-wise defined value $\lambda(\mathbf{x}_i, C)$. This can be interpreted as quantifying the membership strength of $\mathbf{x}_i$ to C. Normalizing these values to the range $[0, 1]$, one may interpret them as probabilities that can be employed for soft clustering. When dealing with data that contains a significant amount of noise, it may happen that the flat clustering obtained via HDBSCAN is too strict and almost every point will end up being an outlier. This is also encountered in this thesis, as discussed in more detail in section 6.6.3.

To conclude this chapter, we want to point out that the improvement of time complexity (as a function of dataset size) is among the biggest advancements that has been made in recent implementations (as employed in this thesis [178]) of the HDBSCAN algorithm.

3.2 Dimensionality reduction

In dimensionality reduction, the goal is to find a compressed representation of the dataset, i.e., to reduce the initial dimensionality d of the features to a value d' with $d' < d$. Ideally, this new set of variables provides the same amount of information as the original data. Reducing the dimensionality of the raw input data can be used as a preprocessing step in a ML problem with a potential boost in performance. Moreover, the visualization of high-dimensional data in a low-dimensional, more comprehensible representation is particularly attractive; for instance, one may construct 2D maps in which data points are distributed in a meaningful way, revealing clustering or clear separation among data points. This way, one can gain insights into complex, high-dimensional representations. In this work, we will compress and study the internal neural-network representations of a given crystal structure (which are 256- and 512-dimensional, depending on the layer – cf. sections 6.2.2 and 6.6.2).

The most common dimensionality reduction method is principal-component analysis (PCA) [182, 183], which determines the directions of highest variance in a dataset. PCA's low-dimensional approximations are linear in nature and thus will struggle to reduce complex high-dimensional data into a lower-dimensional representation that still captures most of the original variance. In this thesis, however, we are especially interested in visualizing the 256- and 512-dimensional neural-network representations for datasets on the order of $10^3 - 10^4$ points. Thus, we need to invoke other, non-linear dimensionality reduction techniques. So-called t-distributed stochastic neighborhood embedding (t-SNE) [184] is a well-known non-linear method, while it has been recently improved in several aspects via the *UMAP* framework [185]. Both UMAP and t-SNE are representatives of so-called manifold learning techniques. These techniques assume that a given data description with a large number of features is artificially high and much less parameters may suffice. Thus, the data points can be perceived as members of a low-dimensional manifold that is embedded in a high-dimensional space [186]. This is also referred to as the *manifold hypothesis*. In general, one may distinguish two types of dimensionality-reduction methods, namely those relying on matrix factorization (such as PCA) and those constructing a neighbor graph (e.g., t-SNE, UMAP).

3.2.1 UMAP

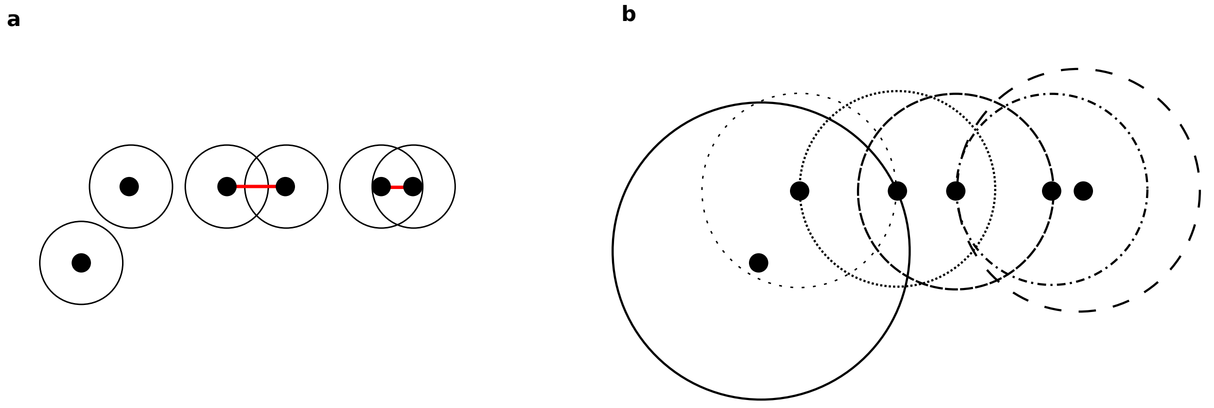

Figure 3.4: Sketch of the neighbor-graph construction that is employed in UMAP for approximating the topological structure of the manifold underlying a given dataset. Here, two points are connected if the associated circles overlap. **a** Connectivity for a uniformly chosen radius. **b** Connectivity for a radius that is adapted for each point. Specifically, the radius is increased until at least k nearest neighbors are contained (here $k = 2$). This corresponds to assigning a different local metric to each point.

The UMAP algorithm models the manifold structure underlying a dataset with a specific topological representation, which in practice turns out to be a weighted graph. The low-dimensional representation of the data is then found by matching the graph of the projection with the graph of the data. The matching procedure is achieved by defining an objective function that quantifies the difference between the two graphs and then minimizing it via stochastic gradient descent. We mainly follow [185] and the associated online documentation (version 0.4.1, https://umap-learn.readthedocs.io/en/latest/how_umap_works.html).

More formally, UMAP finds a so-called fuzzy topological representation of the data, where mathematical tools from algebraic topology [187] and category theory [188] provide theoretical guidance such that this construction captures the topology of the manifold underlying the data. This topological understanding is then used to find a low-dimensional representation (via optimization), which typically is an embedding in 2D Euclidean space – while UMAP also allows to choose different dimensions (e.g., three instead of two embedding-space dimensions) and other, non-Euclidean embedding spaces (e.g., spherical or torus-shaped).

To model the topological structure underlying a given dataset, one can make use of *simplicial complexes*. Loosely speaking, a simplicial complex is obtained by "gluing" together geometric objects, the simplices. A simplex is a k-dimensional object (thus called k-simplex) which is constructed using the convex hull of $k + 1$ independent points. For instance, a 0-simplex would be a point, a 1-simplex a line, a 2-simplex a triangle, and a 3-simplex a tetrahedron. By combining these basic building blocks into simplicial complexes, a multitude of topological spaces can be modeled (as supported by the above-mentioned mathematical machinery).

In UMAP, a specific type of simplicial complex is constructed, the so-called Cěch complex, which relies on the definition of an open cover of the topological space. The open cover can be understood as a collection of sets whose union covers the whole space. Considering the example data set in Fig. 3.4a, the topological space is simply the 2D Euclidean space and the open cover is defined by selecting a certain radius and drawing circles around each point. The simplicial complex would then be constructed by perceiving all sets of the cover as 0-simplices (i.e., points) and then pairwise connecting them via 1-simplices (i.e., lines) if the sets overlap, or connecting three overlapping sets via a 2-simplex etc. While this procedure is not too involved, its abstract formulation (for instance, in terms of simplicial complexes) allows to relate to theorems providing guarantees when this process suffices to represent a given topological space. In practice, to reduce computational cost, it suffices to compute a neighborhood graph (the simplicial complex) only from points and lines (0- and 1-simplices).

One problem is to choose the right criterion – here, the radius – to obtain an open cover, since depending on the selection, different connected components will arise. Given a certain radius, sparsely populated regions may be insufficiently covered while high-density regions lead to the construction of high-dimensional simplices. For uniformly distributed data, these effects could be circumvented by choosing the radius according to the average distance between points. The assumption of uniformly distributed data also makes certain mathematical proofs more easier. While real-world data is not uniformly distributed, UMAP nevertheless makes this assumption, arguing that the lack of uniformity is caused by a varying notion of distance across the manifold. In particular, space stretches in sparse and shrinks in dense regions. For each point, we obtain a different distance function and using Riemannian geometry one can show that this varying notion of distance leads to unit balls (around a given point) being defined as regions that are stretched out to the kth-nearest neighbor. Going back to the example in Fig. 3.4a, this means that for each point we use a different radius, which is determined as the distance to the kth-nearest neighbor. The resulting open cover is illustrated for $k = 2$ in Fig. 3.4b. The choice of k now determines a trade-off between the detection of local, fine details (small k) versus global relationships (large k).

After constructing a k-neighbor graph, one can weight the edges according to the distance (as determined via the local metric). This can be interpreted as the introduction of a fuzzy topology, where each member of the open cover is assigned a (fuzzy) value between zero and one, quantifying its membership. More formally, given an open cover A (here, the set of 1-simplices), one may define the membership function $\mu : A \to [0, 1]$. To provide further intuition, the fact that membership strength decays from the center of the unit ball may be depicted as a blurring effect in the cover. One problem that frequently arises in higher dimensions is the appearance of completely isolated points – the *curse of dimensionality.* To address this, the manifold is assumed to be locally connected, which in practice means that each point should at least be connected to one other point (or rather the members of the open cover extend at least to the first nearest neighbor). In the fuzzy cover this is approached by decaying membership confidence beyond the nearest neighbor.

Note that in general, the local metrics associated to different points $\mathbf{a}$ and $\mathbf{b}$ are not compatible, i.e., the distance from $\mathbf{a}$ and $\mathbf{b}$ is in general different than the distance from $\mathbf{b}$ to $\mathbf{a}$. Practically, the associated weights $w_{\mathbf{a}}, w_{\mathbf{b}}$ are combined as $w_{\mathbf{a}} + w_{\mathbf{b}} - w_{\mathbf{a}} \cdot w_{\mathbf{b}}$, which can be understood as the probability that an edge exists. Applying this unification process, we obtain a fuzzy simplicial complex, which can be interpreted as a weighted k-neighbor graph.

To find a low-dimensional representation, one constructs its fuzzy topological structure in analogy to the procedure explained in the previous paragraphs. If we want to project into a low-dimensional Euclidean space, the manifold structure is much simpler and in particular the metric is constant. In analogy to the procedure described above, we need to define the distance to the nearest neighbors that has previously been computed from the data directly. Thus we have to choose it here as well, introducing a further hyperparameter. The strategy is then to determine the fuzzy topological structure of an initial projected configuration and compare this to the fuzzy topological structure of the data. Comparing these structures requires the introduction of a measure that quantifies the mismatch. In essence, we are comparing weighted graphs in which the weights can be treated as probabilities, indicating the existence of an edge (simplex). The two graphs share the same vertices (0-simplices) and thus one essentially compares probability vectors (indexed by 1-simplices). A natural choice for the objective function is thus the cross entropy (see also Eq. 2.2.1.4).

More formally, a fuzzy set is defined by a reference set A and a function $\mu : A \to [0, 1]$ quantifying the strength of each element $a \in A$. Matching the graphs of original and projected data corresponds to considering two fuzzy sets $(A, \mu_{\text{hdim}}), (A, \mu_{\text{ldim}})$ with differing membership functions but the same set A (here, the 1-simplices or edges). Specifically, for a 1-simplex $a \in A$, $\mu_{\text{hdim}}(a)$ and $\mu_{\text{ldim}}(a)$ correspond to the membership strength in the high and low-dimensional case, respectively. The cross entropy function comparing the two fuzzy sets $(A, \mu_{\text{hdim}}), (A, \mu_{\text{ldim}})$ is then defined as [185]

$$\begin{aligned} C((A, \mu_{\text{hdim}}), (A, \mu_{\text{ldim}})) = \sum_{a \in A} \Bigg(& \mu_{\text{hdim}}(a) \log \left(\frac{\mu_{\text{hdim}}(a)}{\mu_{\text{ldim}}(a)} \right) \\ & + (1 - \mu_{\text{hdim}}(a)) \log \left(\frac{1 - \mu_{\text{hdim}}(a)}{1 - \mu_{\text{ldim}}(a)} \right) \Bigg) . \end{aligned} \tag{3.2.1.1}$$

In practice, minimizing this objective corresponds to the application of attractive and repulsive forces to the graph vertices and edges. Specifically, for a large weight $\mu_{\text{hdim}}(a)$ in the high-dimensional graph, the first term in Eq. 3.2.1.1 is an attractive force between the vertices defining the edge a. The reason is that this term will be minimal if μ_{ldim} is maximized, which is the case when distances in the projection are minimized. For small $\mu_{\text{hdim}}(a)$, the second term in Eq. 3.2.1.1 acts as a repulsive force between the vertices defining a, since this term will be minimal if μ_{ldim} is minimized, which corresponds to maximizing distances in the projection. During optimization, these two forces are balanced and guide the low-dimensional representation into a state that captures the topology underlying the data.

One of the biggest limitations of non-linear dimensionality reduction techniques such as t-SNE and UMAP is that the dimensions of the low-dimensional representations lack interpretability. This is in contrast to methods such as PCA, where the dimensions correspond to directions of greatest variance. UMAP, however, associates to each point a different local metric. Thus, one has to be careful with over-interpreting size of clusters and also their global positioning [189,190]. In particular the choice of hyperparameters can have strong influence on the results. For the applications considered in this work, we obtain reasonable results after grid-searching few values for the number of neighbors, which is denoted as $n_{\text{neighbors}}$. In accordance with other experiments [189], we have to avoid small numbers of neighbors and choose intermediate values (for a dataset of size $\sim$ 8000, a $n_{\text{neighbors}}$ of 100 - 500 suffices to recover the most important structural features in a polycrystal, cf. section 6.2.2). Note that while UMAP shows improved performance in capturing global relationships, the primary goal is to respect the local structure. Thus, if global structure is of particular interest, other techniques may be preferred (e.g., [191] which, however, is more computationally expensive than UMAP). A further parameter is the *minimum distance*, which controls the minimum distance between points in the low-dimensional embedding. For smaller values, points will be more closely packed into clusters, while for larger values, the points appear more spread. In this thesis, the influence of this parameter is studied in Fig. D.4.

Chapter 4

The Bayesian-deep-learning model ARISE

This chapter introduces the crystal-structure recognition framework ARISE. Methods and results presented in this and the following two chapters 5 and 6 follow the lines of our publication

> [122] A. Leitherer, A. Ziletti, and L. M. Ghiringhelli.
> *Robust recognition and exploratory analysis of crystal structures via Bayesian deep learning.*
> Nature Communications **12**, 6234 (2021).
> https://doi.org/10.1038/s41467-021-26511-5

An online tutorial on ARISE can be found at the NOMAD analytics toolkit:

> https://analytics-toolkit.nomad-coe.eu/tutorial-ARISE or alternatively
>
> https://github.com/AndreasLeitherer/Tutorial_ARISE.

Section 4.1 summarizes the information ARISE provides. Details on code and data availability are stated in section 4.2. The necessary steps to arrive at the predictions of ARISE are discussed in section 4.3. This includes preprocessing (section 4.3.1) and materials representation (section 4.3.2), as well as details on the classification model (section 4.3.3). Regarding the model, section 4.3.3.1 explains which architecture we employ while section 4.3.3.2 describes the protocol for generating a sufficiently large training, validation, and test set. Optimization and model performance on pristine and randomly perturbed structures are explained in section 4.3.3.3, while a more detailed discussion on model performance, in particular in comparison with state-of-the-art methods is provided in section 5.1.

4.1 Which information can you obtain from ARISE?

The complete prediction workflow of ARISE is sketched in Fig. 4.1 and described in more detail in section 4.3. For a given input $\mathbf{x}$ (here, the SOAP vector representing the input structure), the model transforms and distributes the information as encoded by SOAP over three hidden layers and one output layer with in total more than 370 000 parameters. The following list explains how useful information can be extracted from each of these parameters. ARISE provides

1. A collection of output samples, which are classification probabilities. The trained model that is considered in this thesis can classify 108 systems (the classes $C := \{c_1, ..., c_{108}\}$). For a given input $\mathbf{x}$, a 108-dimensional output vector $\mathbf{p}$ is computed, whose components correspond to the likelihood of observing class $c_i, i = 1, ..., 108$. Since the model is stochastic, outputs from multiple samplings differ (while it is possible that they are identical or rather highly similar to each other, indicating model confidence – see point 1.2 below). In accordance with the MC dropout framework (cf. section 2.3.2), one samples the output layer multiple times, yielding a collection of probability vectors $\{\mathbf{p}_1, ..., \mathbf{p}_T\}$ for a fixed number T. Values of T on the order of $10^2 - 10^3$ typically suffice to obtain stable predictions. The components of the samples $\mathbf{p}_i$ are referred to as $p(y = c_i|\mathbf{x}, \boldsymbol{\omega}_t)$ in section 2.3, where $\boldsymbol{\omega}_t$ is the set of parameters for a specific forward pass for which a certain number of neurons is dropped. The following information can be extracted from the collected output samples:

    1.1 The classification probabilities assessing the likelihood of each class c_i. This quantity is referred to as $p(y = c_i|\mathbf{x}, D_{\text{train}})$ in section 2.3 and is computed by averaging the output samples (cf. Eq. 2.3.2.1),

    $$p(y = c_i|\mathbf{x}, D_{\text{train}}) \approx \frac{1}{T}\sum_{t=1}^{T} p(y = c|\mathbf{x}, \boldsymbol{\omega}_t). \tag{4.1.0.1}$$

    Note that these averaged quantities already contain more information on model uncertainty than the classification probabilities of the deterministic NN model counterparts (cf. section 2.3 for a more detailed explanation). Two types of information can be inferred from these quantities:

    1.1.1 The predicted class label $\hat{c}$ that corresponds the most likely class. More formally, one determines $\hat{c}$ via

    $$\hat{c} = \underset{c_i \in C}{\text{argmax}}\; p(y = c_i|\mathbf{x}, D_{\text{train}}) \approx \underset{c_i \in C}{\text{argmax}}\; \frac{1}{T}\sum_{t=1}^{T} p(y = c|\mathbf{x}, \boldsymbol{\omega}_t). \tag{4.1.0.2}$$

    1.1.2 A ranking of most similar prototypes, starting with the most likely class as the top prediction.

    1.2 Quantification of model uncertainty. A single number can be calculated from the output samples, where in this work, we employ mutual information (cf. Eq. 2.3.2.5). This way, one can capture and quantify the additional statistical information that is missed by a simple average of the output samples.

2. Internal neural-network representations. The neural network employed in this work consists of three hidden layers (cf. Table 4.1), providing three additional vectors for a given data point: $\mathbf{h}_1 \in \mathbb{R}^{256}, \mathbf{h}_2 \in \mathbb{R}^{512}$, and $\mathbf{h}_3 \in \mathbb{R}^{256}$. As explained in

the previous point, the model is stochastic and thus one actually obtains collections $\{\mathbf{h}_{i,t}\}_{t=1,\dots,T}$ with $i = 1, 2, 3$, which we choose to average, resulting in one vector for each hidden layer, denoted as $\mathbf{h}_i, i = 1, 2, 3$. Given a set of data points, one may inspect the corresponding internal representations to see if the model captures useful relationships, e.g., clusters points in a meaningful way or discovers similarities that are not apparent from the final assignment. We discuss concrete example in sections 6.2.2 and 6.6.2. Regarding the selection of the hidden representations, the results suggest that in principle, each layer can be investigated (cf. Fig. D.1). Intuitively, moving towards the final layer, the model is forced to assign the points to one of the 108 classes. Given the large number of classes, this may result in a significant stretching and bending of internal space, yielding spread out and closely clustered points. Still, depending on the application, hidden representations close to the output may provide useful information.

Both single- and polycrystalline samples can be characterized with this approach. Given an atomic structure, direct application of ARISE provides a global classification, while a local investigation (for instance, of large samples containing grain boundaries) is achieved via *strided pattern matching* (cf. section 6.1). Sensitivity to both periodic and non-periodic systems is established by including corresponding training examples (cf. section 4.3.3.2).

4.2 Code and data availability

In accordance with the principles of reproducible science, all relevant code is provided at https://github.com/angeloziletti/ai4materials as part of the *ai4materials* Python code library while training and test data can be found at https://doi.org/10.5281/zenodo.5526927. For the neural-network part, we employ TensorFlow [192] and Keras [193]), while for SOAP we use the quippy package (https://github.com/libAtoms/QUIP). ai4materials provides a template for integrating various other descriptors as implemented, for instance, in the Dscribe package [194] (https://github.com/SINGROUP/dscribe).

4.3 Prediction workflow

4.3.1 Isotropic scaling

The initial input information is atomic positions and chemical-species symbols (as well as lattice vectors for periodic systems). To reduce the dependency on lattice parameters, we isotropically scale each prototype according to its nearest-neighbor distance d_{NN}. This way, one degree of freedom is eliminated, implying that all cubic systems are equivalent and thus are correctly classified by construction. To compute d_{NN}, we calculate in a first step the histogram of all nearest-neighbor distances. Since the area of spherical shells grows with the squared radius, we divide the histogram by the squared radial distance. Then, we use the center of the maximally populated bin as the nearest-neighbor distance d_{NN}. Dividing the atomic position by d_{NN} yields the final isotropically scaled structure, which is used for calculating the SOAP descriptor. Alternatively, one may use the mean of the nearest

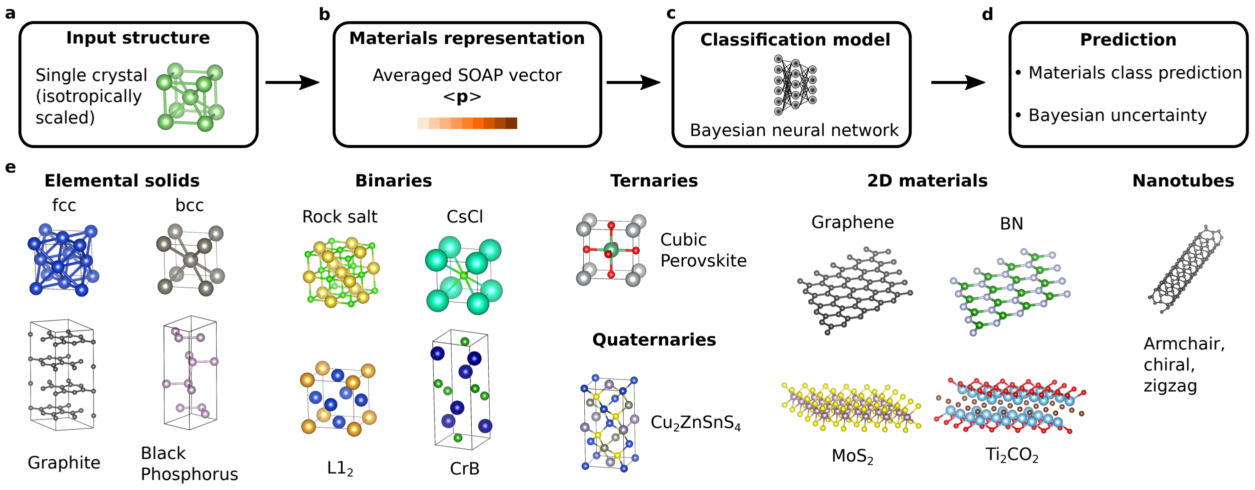

Figure 4.1: Illustration of prediction pipelines for single crystals. In the first step (**a**), the given atomic structure is isotropically scaled according to the nearest-neighbor distance (see section 4.3.1) and then represented via the smooth overlap of atomic position (SOAP) descriptor (**b**, see section 4.3.2). This vectorial materials representation serves as input to the classification model (**c**), which is a Bayesian neural network in this work (cf. section 2.3.2 and 4.3.3). The model is evaluated for multiple forward-passes, giving an ensemble of predictions. Averaging these, and computing the argmax (i.e., determining the entry with maximum probability, cf. Eq. 4.1.0.2) yields the materials class label, while computing the mutual information (cf. Eq. 2.3.2.5) quantifies the uncertainty in the prediction (**d**). **e** Examples of crystallographic prototypes included in the training set. A complete list can be found in the tables 4.2 - 4.4.

neighbors as d_{NN}, which, however, is more prone to defects (e.g., atomic displacements). In case of multiple chemical species, we consider all possible substructures as formed by the constituting species to calculate the SOAP descriptor (see section 4.3.2). For each of the substructures, we compute d_{NN}, while we determine the histogram of neighbor distances only from distances between atoms whose chemical species coincide with those of the substructure. For instance, given the substructure (α, β), i.e., the atomic arrangement of atoms with species β as seen from the perspective of atoms with species α, we consider only α-atoms and determine all distances to β-atoms.

4.3.2 Materials representation

In the following, we provide details on adapting the standard SOAP descriptor such that its number of components is independent on the number of atoms and chemical species.

Starting with the simple case of one chemical species, we consider a local atomic environment $\mathscr{X}$, being defined by a cutoff region (with radius R_{C}) around a central atom, located at the origin of the reference frame. Each atom within this area is represented by a Gaussian function centered at the atomic position $\mathbf{r}_i$ and with width σ. Then, the local atomic density function of $\mathscr{X}$ can be written as [127]

$$\rho_{\mathscr{X}}(\mathbf{r}) = \sum_{i\in\mathscr{X}} \exp\left(-\frac{(\mathbf{r}-\mathbf{r}_i)^2}{2\sigma^2}\right) = \sum_{blm} c_{blm} u_b(r) Y_{lm}(\hat{\mathbf{r}}), \tag{4.3.2.1}$$

where in the second step, an expansion in terms of spherical harmonics $Y_{lm}(\hat{\mathbf{r}})$ and a set of radial basis functions $\{u_b(r)\}$ is performed. One can show that the rotationally invariant power spectrum is given by [127]

$$p(\mathscr{X})_{b_1 b_2 l} = \pi\sqrt{\frac{8}{2l+1}} \sum_m (c_{b_1 lm})^\dagger c_{b_2 lm}. \tag{4.3.2.2}$$

These coefficients can be arranged in a normalized (SOAP) vector $\hat{\mathbf{p}}$ ($\mathscr{X}$), describing the local atomic environment $\mathscr{X}$. In total, we obtain as many SOAP vectors as atoms in the structure, which one can average to obtain a materials descriptor independent of the number of atoms N_{at}. Another possibility (the standard setting in the software we use) is to average the coefficients c_{blm} first and then compute Eq. 4.3.2.2 from this [195]. The cutoff radius R_{C} and σ (cf. Eq. 4.3.2.1) are hyperparameters, i.e., supervised learning cannot be used directly to assign values to these parameters, while their specific choice will affect the results. Typically, one would employ cross-validation while here, we take a different route: First, we assess the similarity between SOAP descriptors using the cosine similarity to identify parameter ranges that provide sufficient contrast between the prototypes. Specifically, we calculate the cross similarity for all pristine prototypes for a range of parameters (e.g., $R_{\mathrm{C}} \in [2.0 \cdot d_{\mathrm{NN}}, 6.0 \cdot d_{\mathrm{NN}}]$ and $[0.05 \cdot d_{\mathrm{NN}}, 0.15 \cdot d_{\mathrm{NN}}]$ for σ). We find that values at and around $\sigma = 0.1 \cdot d_{\mathrm{NN}}$ and $R_{\mathrm{C}} = 4.0 \cdot d_{\mathrm{NN}}$ yield sufficient contrast, i.e., cosine cross-similarities below 1.0. Then, we augment our dataset with SOAP descriptors calculated for different parameter settings that lie within the identified optimal parameter ranges (see section 4.3.3.2).

The extension to several chemical species is achieved by considering all possible substructures as formed by the constituting atoms: Considering NaCl, we first inspect the lattice of

Cl atoms as seen by the Na atoms, which we denote by (Na, Cl); this means that Na atoms are considered as central atoms in the construction of the local atomic environment while only Cl atoms are considered as neighbors. A similar construction is made for the remaining substructures (Na, Na), (Cl, Na), and (Cl, Cl), which may be quite similar depending on the atomic structure. For each substructure, we compute the SOAP vectors via Eq. 4.3.2.2, obtaining a collection of SOAP vectors. Averaging these gives us four (in case of NaCl) averaged SOAP vectors. Averaging the latter again, yields a materials representation being independent on the number of atoms and chemical species.

Formally, given a structure with S species $\alpha_1, ..., \alpha_S$, we consider all substructures formed by pairs of species $(\alpha_i, \alpha_j), j = 1, ..., S$, resulting in S^2 averaged SOAP vectors $< \hat{\mathbf{p}}_{\alpha_i\alpha_j} >_{N_{\text{at},\alpha_i}}$, where the bracket represents the average over number of atoms N_{at} of species α_i. These vectors are averaged over, yielding the final vectorial descriptor $<< \hat{\mathbf{p}}_{\alpha_i\alpha_j} >>_{\alpha_i\alpha_j}$.

Note that this construction of SOAP deviates from the previously reported way [131] of treating multiple chemical species in the following way: Usually, for each atom, one constructs the power spectra [131]

$$p(\mathscr{X})^{\alpha\beta}_{b_1b_2l} = \pi\sqrt{\frac{8}{2l+1}}\sum_m (c^{\alpha}_{b_1lm})^{\dagger} c^{\beta}_{b_2lm}, \tag{4.3.2.3}$$

where the coefficients originate from basis set expansion as in Eq. 4.3.2.2, while the density ρ is constructed separately for each species. For a specific α and β, the coefficients of Eq. 4.3.2.3 can be collected into vectors $\mathbf{p}_{\alpha\beta}$. In case of $\alpha \neq \beta$, cross-correlations, i.e., products of coefficients from different densities are used to construct the vectors $\mathbf{p}_{\alpha\beta}$, which are missing in our version.

4.3.3 The classification model

4.3.3.1 Model architecture

Once the crystal structures are converted into vectors by means of the SOAP representation, we use a neural network model to arrive at a classification decision (cf. Fig. 4.1c). Various neural-network architectures have been developed in recent years [14,196–198]. In a previous work using neural networks for crystal-structure classification [26], the representation was essentially an image, for which convolutional neural networks were used because they produce state-of-the-art results for images. In this work, we employ a fully connected Bayesian neural network (multilayer perceptron), providing classification and uncertainty estimates (cf. Fig. 4.1d). The detailed architecture is reported in Table 4.1. Compared to [26], convolutions are not involved as the input is a vector (cf. section 4.3.2), whose components correspond to basis set expansion coefficients that lack a (spatially) meaningful relationship as it is the case for pixels of an image. The theoretical concepts underlying Bayesian neural networks are described in section 2.3.2. At prediction time, we need to fix T, the number of forward-passes being averaged (cf. Eq. 2.3.2.1). We chose $T = 10^3$ for all results except Fig. 5.1 and 5.3, for which we increased T to 10^5 in order to get stable assignments in case of high uncertainty and very low probability candidates (i.e., $< 1.0\%$). Still, the most similar prototypes can already be determined with 10^3 iterations.

Layer type	Specifications
Input Layer + Dropout	Materials representation (SOAP, 316 components)
Dense Layer + Dropout + ReLU	Size: 256
Dense Layer + Dropout + ReLU	Size: 512
Dense Layer + Dropout + ReLU	Size: 256
Dense Layer + Softmax	Size: 108 (= # classes)

Table 4.1: Architecture of the fully connected Bayesian neural network used in this work. The dropout ratio is 3.17% for all layers. The total number of parameters is 371 820. While training time was fixed to 300 epochs, hyperopt [155] found a batch size of 64 and a learning rate of $2.16 \cdot 10^{-4}$.

4.3.3.2 Training set creation

After both descriptor and model architecture have been identified, a diverse, comprehensive, and materials-science-relevant training set is constructed. The first – and most important – step is to define the structural classes which are going to be included in the model: an overview of the structural classes considered in this work is shown in Fig. 4.1e. A complete list can be found in the tables 4.2 - 4.4. This comprehensive collection of structures includes bulk materials of elemental, binary, ternary, and quaternary composition, as well as 2D materials and carbon nanotubes of chiral, armchair, and zigzag type. In practice, given any database, we extract prototypes, i.e., representative structures that are selected according to some predefined rules. Selection criteria are, for instance, fulfillment of geometrical constraints (number of atoms in the unit cell, number of chemical species) or if the structures are observed in experiment. For the elemental bulk materials, we extract from AFLOW all experimentally observed structures with up to four atoms in the primitive cell. This yields 27 elemental solids encompassing all Bravais lattices, with the exception of monoclinic and triclinic structures because of their low symmetry. Note that this selection includes not only the most common structures such as face-centered-cubic (fcc), body-centered-cubic (bcc), hexagonal-close-packed (hcp), and diamond (which cover more than 80% of the elemental solids found in nature [199]), but also double-hexagonal close-packed (dhcp), graphite (hexagonal, rhombohedral, buckled), and orthorhombic systems such as black phosphorus. This goes already beyond previous work using neural networks for crystal-structure recognition [26], where a smaller set of elemental solids is considered. Also note that multiple close-packings are included [101]: fcc (ABC), hcp (AB), double-hcp (ABAC), and $\alpha-$Sm (ABCBCACAB). For binaries, we select the ten most common binary compounds according to Pettifor [200], plus the $L1_2$ structure because of its technological relevance – for instance, it being the crystal structure of common precipitates in Ni-based superalloys [201]. This selection also includes non-centrosymmetric structure, i.e., structures without inversion symmetry, such as wurtzite. To challenge the classification method with an increasing number of chemical species, a small set of ternary and quaternary materials is included as a proof-of-concept. Specifically, six ternary perovskites [202] (organometal halide cubic and layered perovskites) and six quaternary chalcogenides of A_2BCX_4 type [203] are included due to their relevance in solar cells and photo-electrochemical water splitting devices, respectively.

Going beyond bulk materials, we add an exhaustive set of 46 2D materials [105], comprising not only the well-known elemental structures such as graphene and phosphorene [204] but also binary semiconductors and insulators (BN, GaN), transition metal dichalcogenides (MoS_2), and one example of metal-organic perovskites with six different chemical species. Ternary, quaternary, and 2D materials are taken from the computational materials repository (CMR) [205]. To demonstrate the ability of the proposed framework to deal with complex nanostructures, 12 nanotubes of armchair, chiral and zigzag type are included in the dataset. For each prototype described above, we calculate the SOAP vector with different parameter settings, as is described in the next paragraphs.

To compute the training set (39 204 data points in total), we include periodic and non-periodic systems. For the former, no supercells are necessary (as SOAP is supercell-invariant for periodic structures). For the latter, a given structure (or rather its unit cell as obtained from the respective database) is isotropically replicated until the systems size is at least 100 atoms. Then this supercell structure and the next two larger isotropic replicas are included. With this choice of system sizes, we focus on slab- and bulk-like systems. Note that the network may not generalize to non-periodic structures outside the chosen supercell range. Practically, if the need to classify much smaller or larger supercells arises, one can include additional replicas to the training set and retrain the model (while for larger supercells it is expected that the network will generalize, see also Fig. B.1). Retraining is computationally easy due to fast convergence time. Note that for 2D structures, only in-plane replicas are considered.

In summary, elemental solids and binary compounds are selected from the AFLOW library of crystallographic prototypes [101]. Ternary, quaternary, and 2D materials are taken from the CMR [105, 205]; in particular, we select the parent structure used in the workflow of the reference (e.g., for the graphene class termed "C" in the CMR, we select graphene as representative). Nanotubes are created using the atomic simulation environment (ASE) [206] where the chiral numbers (n, m) provide the class labels. We filter out chiral indices (n, m) (with the integer values n, m taking values in $[0, 10]$) for which the diameter is in the range $[4\,\text{Å}, 6\,\text{Å}]$ (and skipping the cases where $n = m = 0$, $n < m$). Then, we increase the length of each nanotube until at least 100 atoms are contained. No additional lengths are included as it was checked that there is no major change in the SOAP descriptor (via calculating the cosine similarity between descriptors representing nanotubes of different length). For more complex nanotubes (for instance, multi-walled systems), this may change.

Regarding the SOAP parameters, we choose the following range of values: For the cutoff R_C, we select the range $[3.0 \cdot d_{NN}, 5.0 \cdot d_{NN}]$ in steps of $0.2 \cdot d_{NN}$ and for σ the values $[0.08 \cdot d_{NN}, 0.1 \cdot d_{NN}, 0.12 \cdot d_{NN}]$. We calculate the SOAP descriptor using the QUIPPY package (https://libatoms.github.io/QUIP/index.html), where we choose $n_{\max} = 9, l_{\max} = 6$ as limits for the basis set expansion, resulting in an averaged SOAP vector of length 316. Furthermore, we increase the dataset by varying the *extrinsic scaling factor*: For a given prototype, the value of d_{NN} will deviate from the pristine value in presence of defects. Thus, to encode that d_{NN} can have a range of values in certain applications, we scale each pristine prototype not only by $1.0 \cdot d_{NN}$ but also $0.95 \cdot d_{NN}$ and $1.05 \cdot d_{NN}$. We term the factors 0.95, 1.0, 1.05 *extrinsic scaling factors*. One may also see this procedure as a way to increase the training set.

Defective structures are used for testing the model, where in Table 5.1 it is explained how defects (displacements, missing atoms) are introduced. Note that we use the term “missing atoms” and not “vacancies” since most of the percentages of removed atoms we consider are well beyond regimes found in real materials. Also note that displacements as high as 4% of the nearest-neighbor distance might already cause a transition to the liquid phase in some solids. Still, as noted in the introduction in section 1.3, experimental and computational data often present levels of distortions which are comparable or even substantially exceed these regimes. We introduce defects for all pristine prototypes included in the training set (specifically, for the supercells determined as described in the first paragraph – for both periodic and non-periodic boundary conditions, while for nanotubes only non-periodic structures are used). Since the defects are introduced randomly, we run 10 iterations of defect creation on each prototype. Then, we calculate SOAP for all of these defective structures for one specific parameter setting ($R_{\mathrm{C}} = 4.0 \cdot d_{\mathrm{NN}}$, $\sigma = 0.1 \cdot d_{\mathrm{NN}}$, extrinsic scaling factor= 1.0), which corresponds to the center of the respective parameter ranges included in the training set. Finally, we obtain 5880 defective structures for each defect ratio. In total, we compute defective structures for three defect types (missing atoms and displacements introduced both separately and combined) for eight different defect ratios, giving in total 141,120 defective data points.

Note that the (highly) defective structures are test structures while training is performed using pristine structures only. The main reason is that in general, by introducing high levels of random defects, a structure may be more similar to another prototype than the intended parent prototype and thus the label given to the model during training would have to be adapted.

#	Prototype	Symmetry	Material type	Data source
1.	bcc (W)	229, cubic	Bulk, Elemental	AFLOW / NOMAD
2.	diamond (C)	227, cubic	Bulk, Elemental	AFLOW / NOMAD
3.	fcc (Cu)	225, cubic	Bulk, Elemental	AFLOW / NOMAD
4.	α-Po	221, (simple) cubic	Bulk, Elemental	AFLOW / NOMAD
5.	hcp (Mn)	194, hexagonal	Bulk, Elemental	AFLOW / NOMAD
6.	α-La (dhcp)	194, hexagonal	Bulk, Elemental	AFLOW / NOMAD
7.	Hex. diamond	194, hexagonal	Bulk, Elemental	AFLOW / NOMAD
8.	Hex. graphite	194, hexagonal	Bulk, Elemental	AFLOW / NOMAD
9.	Sn	191, (simple) hexagonal	Bulk, Elemental	AFLOW / NOMAD
10.	Buckled graphite	186, hexagonal	Bulk, Elemental	AFLOW / NOMAD
11.	α-As	166, rhombohedral	Bulk, Elemental	AFLOW / NOMAD
12.	α-Hg	166, rhombohedral	Bulk, Elemental	AFLOW / NOMAD
13.	α-Sm	166, rhombohedral	Bulk, Elemental	AFLOW / NOMAD
14.	β-O	166, rhombohedral	Bulk, Elemental	AFLOW / NOMAD
15.	β-Po	166, rhombohedral	Bulk, Elemental	AFLOW / NOMAD
16.	γ-Se	152, trigonal hexagonal	Bulk, Elemental	AFLOW / NOMAD
17.	Rhomb. graphite	166, rhombohedral	Bulk, Elemental	AFLOW / NOMAD
18.	α-Pa	139, (body-centered) tet.	Bulk, Elemental	AFLOW / NOMAD
19.	β-Sn	141, (body-centered) tet.	Bulk, Elemental	AFLOW / NOMAD
20.	In	139, (body-centered) tet.	Bulk, Elemental	AFLOW / NOMAD
21.	γ-N	136, (simple) tet.	Bulk, Elemental	AFLOW / NOMAD
22.	β-Np	129, (simple) tet.	Bulk, Elemental	AFLOW / NOMAD
23.	γ-Pu	70, (face-centered) orth.	Bulk, Elemental	AFLOW / NOMAD
24.	α-Ga	64, (base-centered) orth.	Bulk, Elemental	AFLOW / NOMAD
25.	Black phosphorus	64, (base-centered) orth.	Bulk, Elemental	AFLOW / NOMAD
26.	Molecular iodine	64, (base-centered) orth.	Bulk, Elemental	AFLOW / NOMAD
27.	α-U	63, (base-centered) orth.	Bulk, Elemental	AFLOW / NOMAD
28.	NaCl	225, cubic	Bulk, Binary	AFLOW / NOMAD
29.	CsCl	221, cubic	Bulk, Binary	AFLOW / NOMAD
30.	$L1_2$ (Cu_3Au)	221 (simple) cubic	Bulk, Binary	AFLOW / NOMAD
31.	Zinc blende (ZnS)	216, (face-centered) cubic	Bulk, Binary	AFLOW / NOMAD
32.	FeSi	198 (simple) cubic	Bulk, Binary	AFLOW / NOMAD
33.	NiAs	194, hexagonal	Bulk, Binary	AFLOW / NOMAD
34.	Wurtzite (ZnS)	186, hexagonal	Bulk, Binary	AFLOW / NOMAD
35.	$L1_0$ (CuAu)	123, (simple) tet.	Bulk, Binary	AFLOW / NOMAD
36.	CrB	63, (base-centered) orth.	Bulk, Binary	AFLOW / NOMAD
37.	MnP	62, (simple) orth.	Bulk, Binary	AFLOW / NOMAD
38.	FeB	62, (simple) orth.	Bulk, Binary	AFLOW / NOMAD
39.	$AgNbO_3$	cubic	Bulk, Ternary	CMR
40.	$CsSnI_3$	cubic	Bulk, Ternary	CMR
41.	$CsSnCl_3$	tetragonal	Bulk, Ternary	CMR
42.	Cs_2WO_4	tetragonal	Bulk, Ternary	CMR
43.	$Ca_3Ge_2O_7$	tetragonal	Bulk, Ternary	CMR
44.	$CsSnCl_3$	orthorhombic	Bulk, Ternary	CMR

Table 4.2: Complete list of prototypes (part I) included in the training set of this work. If provided by the respective resources, information on space group, crystal system or Bravais lattice is listed.

#	Prototype	Symmetry	Material type	Data source
45.	$Cu_2BaGeSe_4$	144 (trigonal)	Bulk, Quaternary compound	CMR
46.	Cu_2CdSnS_4	121 (tetragonal)	Bulk, Quaternary compound	CMR
47.	Cu_2ZnSnS_4	82 (tetragonal)	Bulk, Quaternary compound	CMR
48.	Cu_2KVS_4	40 (orthorhombic)	Bulk, Quaternary compound	CMR
49.	Cu_2CdGeS_4	31 (orthorhombic)	Bulk, Quaternary compound	CMR
50.	Cu_2ZnSiS_4	7 (monoclinic)	Bulk, Quaternary compound	CMR
51.	Graphene	191 (hexagonal)	2D Materials	CMR
52.	Ti_3C_2	187 (hexagonal)	2D Materials	CMR
53.	$Ti_3C_2O_2$	187 (hexagonal)	2D Materials	CMR
54.	MoS_2	187 (hexagonal)	2D Materials	CMR
55.	$Ti_3C_2H_2O_2$	187 (hexagonal)	2D Materials	CMR
56.	GaS	187 (hexagonal)	2D Materials	CMR
57.	BN	187 (hexagonal)	2D Materials	CMR
58.	$Ti_2CH_2O_2$	164 (trigonal)	2D Materials	CMR
59.	Ti_2CO_2	164 (trigonal)	2D Materials	CMR
60.	CdI_2	164 (trigonal)	2D Materials	CMR
61.	CH	164 (trigonal)	2D Materials	CMR
62.	CH_2Si	156 (trigonal)	2D Materials	CMR
63.	Ti_4C_3	156 (trigonal)	2D Materials	CMR
64.	BiTeI	156 (trigonal)	2D Materials	CMR
65.	$Ti_4C_3O_2$	156 (trigonal)	2D Materials	CMR
66.	GeSe	156 (trigonal)	2D Materials	CMR
67.	MoSSe	156 (trigonal)	2D Materials	CMR
68.	$Ti_4C_3H_2O_2$	156 (trigonal)	2D Materials	CMR
69.	$AgBr_3$	150 (trigonal)	2D Materials	CMR
70.	$TiCl_3$	150 (trigonal)	2D Materials	CMR
71.	BiI_3	147 (trigonal)	2D Materials	CMR
72.	FeSe	129 (tetragonal)	2D Materials	CMR
73.	PbSe	123 (tetragonal)	2D Materials	CMR
74.	GeS_2	115 (tetragonal)	2D Materials	CMR
75.	C_3N	65 (orthorhombic)	2D Materials	CMR
76.	FeOCl	59 (orthorhombic)	2D Materials	CMR
77.	P	28 (orthorhombic)	2D Materials	CMR
78.	PdS_2	14 (monoclinic)	2D Materials	CMR
79.	MnS_2	14 (monoclinic)	2D Materials	CMR
80.	GaSe	12 (monoclinic)	2D Materials	CMR
81.	TiS_3	11 (monoclinic)	2D Materials	CMR
82.	WTe_2	11 (monoclinic)	2D Materials	CMR
83.	HfBrS	7 (monoclinic)	2D Materials	CMR
84.	RhO	6 (monoclinic)	2D Materials	CMR
85.	SnS	6 (monoclinic)	2D Materials	CMR
86.	NiSe	6 (monoclinic)	2D Materials	CMR
87.	AuSe	6 (monoclinic)	2D Materials	CMR

Table 4.3: Complete list of prototypes (part II) included in the training set of this work.

#	Prototype	Symmetry	Material type	Data source
88.	VTe_3	6 (monoclinic)	2D Materials	CMR
89.	ReS_2	2 (monoclinic)	2D Materials	CMR
90.	$ScPSe_3$	1 (triclinic)	2D Materials	CMR
91.	PbA_2I_4	1 (triclinic)	2D Materials	CMR
92.	PbS	1 (triclinic)	2D Materials	CMR
93.	CrW_3S_8	1 (triclinic)	2D Materials	CMR
94.	$VPSe_3$	1 (triclinic)	2D Materials	CMR
95.	$CrWS_4$	1 (triclinic)	2D Materials	CMR
96.	$MnPSe_3$	1 (triclinic)	2D Materials	CMR
97.	CNT	armchair, (3,3), $30.0°, 4.07$ Å	Nanotubes, mono-species	ASE
98.	CNT	armchair, (4,4), $30.0°, 5.42$ Å	Nanotubes, mono-species	ASE
99.	CNT	chiral, (4,2), $19.11°, 4.14$ Å	Nanotubes, mono-species	ASE
100.	CNT	chiral, (4,3), $25.28°, 4.76$ Å	Nanotubes, mono-species	ASE
101.	CNT	chiral, (5,1), $8.95°, 4.36$ Å	Nanotubes, mono-species	ASE
102.	CNT	chiral, (5,2), $16.1°, 4.89$ Å	Nanotubes, mono-species	ASE
103.	CNT	chiral, (5,3), $21.79°, 5.48$ Å	Nanotubes, mono-species	ASE
104.	CNT	chiral, (6,1), $7.59°, 5.13$ Å	Nanotubes, mono-species	ASE
105.	CNT	chiral, (6,2), $13.9°, 5.65$ Å	Nanotubes, mono-species	ASE
106.	CNT	chiral, (7,1), $6.59°, 5.91$ Å	Nanotubes, mono-species	ASE
107.	CNT	zigzag, (6,0), $0.0°, 4.7$ Å	Nanotubes, mono-species	ASE
108.	CNT	zigzag, (7,0), $0.0°, 5.48$ Å	Nanotubes, mono-species	ASE

Table 4.4: Complete list of prototypes (part III) included in the training set of this work. For the carbon nanotubes (CNTs), the symmetry column specifies the configuration type (chiral, zigzag or armchair) together with the corresponding chiral numbers (n, m), the chiral angle θ and the nanotube diameter.

4.3.3.3 Optimization

Training is performed using Adam optimization [10] (see also section 2.2.2 for more details). The multilayer perceptron is implemented in Keras [193] using TensorFlow [192] as backend. Furthermore we optimize hyperparameters such as the number of layers using Bayesian optimization, specifically the Tree-structured Parzen estimator (TPE) algorithm as provided by the Python library hyperopt [155] (see also section 2.2.4 for more details). Only pristine structures are used for training. Bayesian optimization (TPE) provides a list of candidate models, from which we select the most robust one and report its architecture in Table 4.1. More details are provided in the following.

The initial training set is split (80/20% training / validation split of pristine structures, performed using scikit-learn function and random state of 42) and the accuracy on the validation set is used as the performance metric to be minimized via hyperopt (for 50 iterations). Fast convergence (followed by oscillations around high-accuracy values) or divergence is typically observed, which is why we train for a fixed number of epochs (300) and save only the model with the best performance on the validation set. This is in accordance with the early stopping procedure (cf. section 7.8 in [14]) that is discussed in the last paragraph of section 2.2.3. Training is performed on 1 GPU (Tesla Volta V100 32GB) on the Talos machine-learning cluster at the Max Planck Computing and Data Facility (MPCDF). We observe that accuracies around 99% can be reached after few iterations, with individual training runs converging within 20 minutes, depending on model complexity.

Practically, strong models are obtained if training is done only on pristine structures but further fine-tuning can be made to reach perfect accuracies. First, we restrict to one setting of training parameters (see previous section). From a computational efficiency point of view, this is also the preferred choice since one has to compute only one descriptor per structure during prediction time. We select $R_{\mathrm{C}} = 4.0 \cdot d_{NN}$ and $\sigma = 0.1 \cdot d_{\mathrm{NN}}$ as well as an extrinsic scaling factor of 1.0. These choices are at the center of the respective parameter ranges. While the model with highest validation accuracy (on the whole training set) determined via hyperopt usually gives very strong performance, it is not necessarily the best possible one, especially in terms of generalization ability to defective structures. To find the optimal (i.e., most robust) model we select some of the best models (e.g., top 15) found via hyperopt and rank them based on their performance on pristine and defective structures (again for one setting of R_{C}, σ). In particular, we restrict to defective points with either $\leq 5\%$ atoms missing or $< 1\%$ atomic displacement, which comprises 35 280 data points (six different defect ratios with 5 880 points each). The number of pristine data points is 396. Using this strategy, we can identify a model with 100% accuracy on pristine and defective structures, which is reported in the last line of Table 5.1. The accuracy on the whole training set comprising 39 204 data points is 99.66%.

We also investigate the performance on higher defect ratios beyond physically reasonable perturbations, since this is typically encountered in atom-probe experiments. In particular, we investigate three defect types (missing atoms, displacements, and both of them) comprising 105 840 data points. The results for missing atoms ($> 5\%$) and displacements ($> 0.6\%$) can be found in the last line of Tables 5.1 and 5.2. Classification accuracies on structures with both missing atoms and displacements are specified in the last line of Table 5.3. Note that training and model selection only on pristine structures can yield robust models, especially if the number of classes is reduced. For instance, training only on binary systems using a pristine set of 4 356 data points (full SOAP parameter range) gives perfect accuracy on both the full training set and 3 960 defective structures (displacements $\leq 0.06\%$ and $\leq 5\%$ missing atoms – for the setting $R_{\mathrm{C}} = 4.0 \cdot d_{\mathrm{NN}}, \sigma = 0.1 \cdot d_{\mathrm{NN}}$, extrinsic scaling factor 1.0). Note that in general, if fewer classes are considered (e.g., ~ 20), the training time can be significantly reduced (e.g., to a few minutes).

Chapter 5

Global crystal characterization using ARISE

The following sections contain several applications of ARISE for the global analysis of crystal structures, i.e., the assignment of a single label to a list of atomic positions and chemical species symbols. First, we benchmark our method with respect to state-of-the-art methods, using a comprehensive synthetic dataset of pristine and (heavily) defective single crystals (cf. section 5.1). In section 5.2, we apply ARISE to data originating from a completely different resource, in particular experimental STEM images of graphene. Then, predictions and uncertainty of ARISE are analyzed for continuous structural transformations in section 5.3. Specifically, the so-called Bain path is investigated, which is relevant for several applications (e.g., steel properties). Moreover, the Bain path describes structural transitions similar to the ones taking place in the electron tomography data that is analyzed in section 6.6.1. Finally, a study of out-of-sample structures is provided in section 5.4, where ARISE is faced with structures it has never encountered during training (and that are not randomly distorted versions of the training structures, i.e., this analysis is qualitatively different from the one in section 5.1).

5.1 Benchmarking

We first compare ARISE's performance on pristine and defective structures with state-of-the-art crystal-structure recognition methods, specifically spglib [111], polyhedral template matching (PTM) [116], common neighbor analysis (CNA) [113], adaptive common neighbor analysis (a-CNA) [114], and bond angle analysis (BAA) [115] (cf. Table 5.1). None of the benchmarking methods can treat all the materials shown in Fig. 4.1e; thus for fairness, the classification accuracy is only calculated for classes for which the respective methods were designed for, implying that most structures are excluded. Including all 108 classes will lead to reduced accuracy scores which can be found in the tables A.1, A.2, and A.3.

We want to emphasize that ARISE and the above mentioned methods have several conceptual differences that require specific choices in order to establish a fair benchmarking protocol. Besides the strong disparity in terms of classifiable systems, methods such as PTM, CNA, a-CNA, and BAA provide a per-atom classification. From these assignments, we have to calculate a single accuracy value to enable a comparison on equal footing. Moreover, spglib is restricted to predicting the space group, hindering its application to non-periodic systems or structures where no symmetry can be detected (i.e., the space group is 1). In the following, we concentrate on the analysis of the classification accuracies while coming back to a detailed discussion of our choices at the end of this section.

The performance on pristine structures is reported in Table 5.1. The accuracy in classifying pristine structures is always 100% as expected, with the only exception being CNA: For this method, the default cutoff only allows to correctly classify fcc and bcc but not hcp structures. For defective structures, the situation is drastically different. Spglib classification accuracy on displaced structures is low, and only slightly improved by using loose setting (up to 1% displacement). For missing atoms, the accuracy is very low already at the 1% level regardless of the setting used. Note, however, this is actually spglib's desired behavior since the aim of this method is not robust classification. Moreover, the tolerance is in place mainly to account for truncation errors in the storage of coordinates. As indicated in the first column of Table 5.1, spglib can treat 96 out of the 108 prototypes included in our dataset with the twelve missing prototypes being carbon nanotubes. Methods based on local atomic environment (PTM, BAA, CNA, a-CNA) perform very well on displaced structures, but they suffer from a substantial accuracy drop for missing-atoms ratios beyond 1%. Their biggest drawback, however, is that they can treat only a handful of classes: three classes for BAA, CNA, and a-CNA, and twelve classes for PTM. ARISE is very robust with respect to both displacements and missing atoms (also for higher levels of defects, cf. Table 5.2, and even concurrently, cf. Table 5.3), while being the only method able to treat all 108 classes included in the dataset, including complex systems, such as carbon nanotubes. An uncertainty value quantifying model confidence is also returned, which is particularly important when investigating defective structures or inputs that are far out of the training set. Section 5.2 provides an application (STEM images of graphene) that highlights the correlation of the predictive uncertainty with the level of distortion. For out-of sample structures, we provide a detailed study in section 5.4 where we challenge ARISE with structures it has not been trained on, i.e., it is forced to fail by construction. We find that ARISE returns non-trivial physically meaningful predictions, thus making it particularly attractive, e.g., for screening large and structurally diverse databases. Moreover, we analyze predictions and uncertainty of ARISE for continuous structural transformations in section 5.3, where we consider the so-called Bain path that includes transitions between fcc, bcc, and tetragonal structures. In this application, both out-of-sample and (uniformly) distorted structures appear. Note that these applications refer to the global analysis of atomic structures, while we present several further examples for the local study of large, polycrystalline samples in section 6, including the detection of grain boundaries (cf. Fig. 6.2, 6.6) or precipitates (cf. Fig. 6.4), as well as the local analysis of nanoparticles (cf. Fig. 6.8, 6.10). Also in these settings the predictive uncertainty turns out to correlate with the level of defectiveness and yields reasonable results for data points far outside the training set (e.g., at grain boundaries where two phases are mixed).

In the following, we provide additional details, in particular how the classification accuracy is calculated for each method and which structures from the training set are included to test spglib, PTM etc. :

	Pristine	Random displacements (δ)					Missing atoms (η)			
		0.1%	0.6%	1%	2%	4%	1%	5%	10%	20%
Spglib (loose, 96 / 108)	100.00	100.00	100.00	95.26	20.00	0.00	11.23	0.00	0.00	0.00
Spglib (tight, 96 / 108)	100.00	0.00	0.00	0.00	0.00	0.00	11.23	0.00	0.00	0.00
PTM (12 / 108)	100.00	100.00	100.00	100.00	100.00	100.00	88.67	51.76	25.93	6.24
CNA (3 / 108)	66.14	62.81	62.81	54.55	32.34	31.41	55.86	32.50	15.75	3.07
a-CNA (3 / 108)	100.00	100.00	100.00	100.00	100.00	100.00	89.25	52.81	25.92	5.37
BAA (3 / 108)	100.00	100.00	100.00	100.00	100.00	97.85	99.71	88.78	65.21	25.38
ARISE (108 / 108)	100.00	100.00	100.00	100.00	99.86	99.29	100.00	100.00	100.00	99.85

Table 5.1: **Accuracy in identifying the parent class of defective crystal structures.** The defective structures are generated by randomly displacing atoms according to a uniform distribution on an interval $[-\delta \cdot d_{\mathrm{NN}}, +\delta \cdot d_{\mathrm{NN}}]$ proportional to the nearest-neighbor distance d_{NN} (central panel), or removing η% of the atoms (right panel). The accuracy values shown are in percentage. For benchmarking, we use Spglib [111] (with two settings for the precision parameters, "loose" (position/angle tolerance 0.1Å/ 5°) and "tight" (position/angle tolerance 10^{-4} / 1°)), polyhedral template matching (PTM) [116], common neighbor analysis (CNA) [113], adaptive common neighbor analysis (a-CNA) [114], and bond angle analysis (BAA) [115]. The number of classes which can be treated out of the materials pool in Fig. 4.1e is shown in parentheses for each method. Spglib can assign a space group to all materials except the 12 nanotubes. PTM can only classify 7 elemental and 5 binary materials of those considered in this work. Additional classes are missing for CNA, a-CNA, and BAA as they cannot classify simple cubic (sc) and diamond structures. The approach proposed here can be applied to all classes, and thus the whole dataset is used (see Tables 4.2-4.4 for a complete list).

	Random displacements (δ)		Missing atoms (η)	
	7%	10%	25%	30%
Spglib (loose)	0.00	0.00	0.00	0.00
Spglib (tight)	0.00	0.00	0.00	0.00
PTM	100.00	94.34	3.33	1.72
CNA	31.41	24.20	1.38	0.55
a-CNA	99.99	94.55	2.60	1.03
BAA	87.79	69.68	14.25	7.35
ARISE (this work)	97.82	94.56	99.86	99.76

Table 5.2: Accuracy in identifying the parent class of defective crystal structures for high displacements (percentage δ) and missing atoms (percentage η).

- For spglib, we only include prototypes from AFLOW. The reason for excluding structures from the computational materials repository "CMR" is that we do not always have the correct or meaningful labels for all structures. For instance, some 2D materials are specified as P1 in the database, which cannot be used as a correct label. Furthermore, for quaternary chalcogenides, the expected symmetries (as specified in the corresponding reference [203]) cannot be reconstructed, which is most likely due to local optimization effects. Similar observations were made for the ternary perovskites. More careful choice of precision parameters or additional local optimization may help. Thus, to enable a fair comparison, the benchmarking only reports results on elemental and binary compounds from AFLOW (where we know the true labels), while the performance on all data is shown in table A.1, A.2, and A.3. To avoid the impression that spglib is not applicable to ternary, quaternary, and 2D materials, we still provide the label "96/108" behind spglib methods in the benchmarking tables. Note that non-periodic structures (supercells and carbon nanotubes) are excluded, since these systems cannot be treated by spglib.

- For the other benchmarking methods, which are common neighbor analysis (CNA, a-CNA), bond angle analysis (BAA), and polyhedral template matching (PTM), we use implementations provided in OVITO [207], where for BAA we apply the Ackland Jones method. As for spglib, only periodic structures were included. BAA, CNA, a-CNA all include fcc, bcc, and hcp structures, while PTM contains in addition sc, diamond, hexagonal diamond, graphene, graphitic boron nitride, $L1_0$, $L1_2$, zinc blende, and wurtzite. Each of the frameworks provide one label for each atom, i.e., for a structure with N atoms we obtain N labels. To obtain an accuracy score, we compare these N predictions to N true labels, which correspond to the space group associated with the prototype label (e.g., 194 for hcp). For CNA, we select the standard cutoff (depending on its value one is able to detect bcc but not hcp and vice versa). Also for BAA (Ackland Jones) and a-CNA standard settings are used. For PTM, an RMSD cutoff of 0.1 was used (again default in OVITO). Note that PTM can also distinguish different sites of the $L1_2$ structure. For simplicity, we did not label the $L1_2$ structure by sites

	Missing atoms and displacements (η, δ)						
	(1%, 0.1%)	(5%, 0.6%)	(10%, 1%)	(15%,2%)	(20%,4%)	(25%,7%)	(30%,10%)
Spglib (loose)	11.32	0.00	0.00	0.00	0.00	0.00	0.00
Spglib (tight)	0.00	0.00	0.00	0.00	0.00	0.00	0.00
PTM	88.68	51.78	25.60	12.75	6.41	3.19	1.46
CNA	55.77	31.95	13.83	4.41	2.03	0.79	0.19
a-CNA	89.21	52.36	26.01	12.13	6.07	2.40	0.97
BAA	99.72	88.98	65.17	42.62	25.95	15.58	6.63
ARISE (this work)	100.00	100.00	100.00	99.88	99.29	97.31	92.50

Table 5.3: Accuracy in identifying the parent class of defective crystal structures, with both missing atoms (percentage η) and displacements (percentage δ) introduced at the same time. The results show that ARISE is also robust for highly defective structures where displacements and missing atoms are present at the same time. This is the typical situation encountered in APT – making ARISE arguably the best available candidate for crystal-structure classification in APT experimental data.

and take this classification into account, but always assigning a true label as soon as an atom was assigned to the $L1_2$ class (even if it might be not the correct site).

- Furthermore, for ARISE periodic and non-periodic structures are included, while for the benchmarking methods only periodic structures are considered. While for spglib, translational symmetry is violated by construction, the other methods can in principle be applied to these systems. However, when calculating the accuracy for a given non-periodic structure, we have to choose a label for the boundary atoms. If we select the same label for these atoms as for the central ones (which have a sufficiently larger number of neighbors), these methods will usually predict the class "None" and interpreting this as a misclassification would decrease the total classification accuracy. Therefore, for a fair comparison, we exclude non-periodic structures.

5.2 Application to STEM experimental images

So far, we have tested ARISE on highly defective structures derived from pristine crystallographic prototypes. We now investigate defective structures originating from a completely different data source, namely STEM experiments, to demonstrate the generalization ability of ARISE and its applicability to experimental data. STEM experiments are a valuable resource to characterize material specimens, and to study, for instance, the atomic structure at grain boundaries [98]. Atomic resolution can be reached in high-angle annular dark-field (HAADF) images.

To test ARISE, we consider the two experimental HAADF images of graphene shown in Fig. 5.1a; these images contain a substantial amount of noise which makes it very challenging to recognize the graphene honeycomb pattern by naked eye. The choice of graphene is motivated by it being one of the few completely flat 2D materials; x and y atomic positions obtained from STEM images thus provide the actual crystal structure, and not a mere projection. The first step is to obtain approximate atomic positions from the HAADF experimental images. To this end, we employ the deep-learning framework AtomNet [81] based on fully convolutional neural networks [208]. These are essentially variants of convolutional neural networks in which no fully connected neural network is employed after the convolutional layers. For AtomNet, given an input image, the output is a pixel matrix of the same size with each entry corresponding to the probability that the associated pixel contains an atom. We refer to section chapter 9 of [14] and our online tutorial (https://github.com/AndreasLeitherer/Tutorial_CNN) for an introduction to convolutional neural networks. Fully convolutional neural networks are reviewed, for instance, in section 13.11 of [52]. The inferred atomic positions (i.e., x and y coordinates) are shown in Fig. 5.1b. ARISE is then used to classify the structures shown in Fig. 5.1b following the steps being summarized in Fig. 4.1.

The top predictions ranked by classification probability are shown in Fig. 5.1d, together with the uncertainty of the assignments as quantified by the mutual information. ARISE correctly recognizes both images as being graphene, despite the substantial amount of noise present in the images and reconstructed atomic positions. For the first image (Fig. 5.1a-b top), ARISE predicts graphene with very high probability ($\sim 99\%$). Indeed, the similarity to graphene is apparent, although evident distortions are present in some regions (for instance, bonds which are expected to be aligned are clearly displaced – see the two dashed ellipses in Fig. 5.1b, top). The second candidate structure is C_3N, predicted with $\sim 1\%$ probability; in C_3N, nitrogen atoms are arranged in a graphene lattice, making also this low probability assignment physically meaningful. (Note that the materials representation is obtained as average of substructures, see section 4.3.2 for more details).

For the second image (Fig. 5.1a-b bottom), ARISE also correctly predicts graphene, this time with 79% probability. The uncertainty is six times larger than in the previous case. Indeed, this structure is much more defective than the previous one: it contains a grain boundary in the lower part, causing evident deviations from the pristine graphene lattice, as illustrated by the vertical dashed line and the dashed red circle in Fig. 5.1a-b bottom. The other four candidate structures appearing in the top five predictions (PbSe, MnS_2, BN, C_3N in decreasing order of classification probability) are the remaining completely flat monolayer structures that are known to the neural network (out of the 108 structures in the training dataset, only five are flat monolayers). Note that no explicit information about the dimensionality of the material is given to the model. In particular, a given input is compared to 108 different crystallographic prototypes.

It is important to point out that ARISE robustness well beyond physical levels of noise (cf. Tables 5.1-5.3) is essential to achieve the correct classification despite the presence of substantial amount of noise from both experiment and atomic position reconstruction. Moreover, the uncertainty (i.e., mutual information) allows to quantify how defective a structure is; thus, the presented procedure of combining reconstruction methods (here AtomNet) with the crystal-structure classification model ARISE may be used to automatically evaluate the quality of STEM images. In particular, one may automatically screen for STEM images that are likely to contain defects such as grain boundaries.

5.3 The Bain transformation path

The Bain transformation path describes structural transitions between bcc and fcc symmetries via intermediate tetragonal phases [209] of body-centered – or equivalently – face-centered tetragonal symmetry. Originally investigated for iron [209], the Bain path is relevant in thermo-mechanical processing – a central aspect for steel properties [210] – as it serves as a model for temperature-induced transitions between fcc (γ) and bcc (α) iron [211]. The Bain path is also crucial for understanding properties of epitaxial films [212, 213] or metal nanowires [214].

Practically, the structures constituting a Bain path can be obtained by varying the ratio c/a between lattice parameters a and c of a tetragonal structure (cf. Fig. 5.2a); $c/a = 1$ corresponds to a cubic structure. We generate tetragonal geometries for lattice parameters a, c taking values in $[3.0\,\text{Å}, 6.0\,\text{Å}]$ with steps of $0.05\,\text{Å}$, resulting in 3 721 crystal structures. These structures are then classified with ARISE, and the results depicted via classification and uncertainty maps in Fig. 5.2b and c, respectively. Each point in these maps corresponds to a prediction for a specific geometry. We include in the training set fcc, bcc, and tetragonal geometries with structural parameters known experimentally; they are shown as stars in Fig. 5.2b. Specifically, the lattice parameters $(a, c, c/a)$ are $(3.155\text{Å}, 3.155\text{Å}, 1.0)$ for the bcc [215]

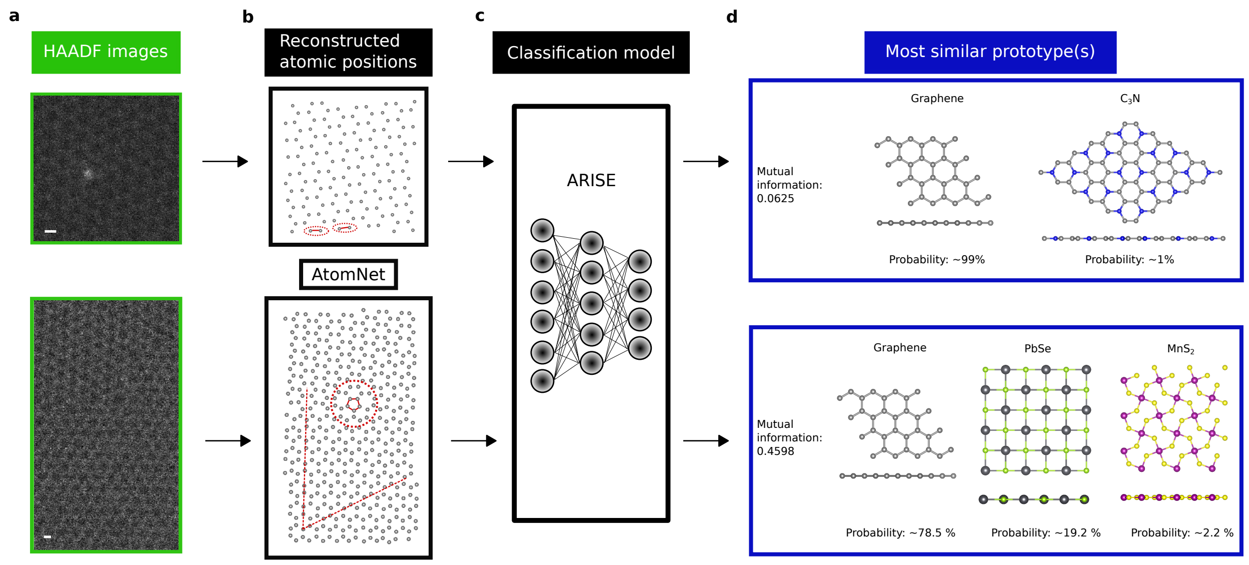

Figure 5.1: **a** Experimental HAADF images (from https://github.com/pycroscopy/AICrystallographer/tree/master/AtomNet) of two graphene structures. White scale bars in the bottom left of the images correspond to the typical graphene bond length (1.42 Å). Atomic positions are reconstructed from these images via AtomNet [81] (**b**). The resulting atomic structures are analyzed using ARISE (**c**). The top predicted structures are visualized and mutual information is used to quantify the uncertainty in the classification (**d**).

and $(3.615\text{Å}, 5.112\text{Å}, \sqrt{2})$ for the fcc prototype [216], while two tetragonal structures (being assigned one common label "tetragonal") are included with $(3.253\text{Å}, 4.946, \text{Å}, 1.521)$ in case of In [217] and $(3.932\text{Å}, 3.238\text{Å}, 0.824)$ for $\alpha-$Pa [218]. We isotropically scale every geometry to remove one degree of freedom (cf. section 4.3.1) , so that all possible cubic lattices are effectively equivalent; this allows the model to generalize by construction to all cubic lattices regardless of the lattice parameter. The same holds for tetragonal structures (i.e., two degrees of freedom) with constant c/a ratio. As visual aid, we mark lines of constant c/a in Fig. 5.2b-c starting from the four structures included in the training set. Note that any path connecting the constant c/a ratios corresponding to fcc and bcc structures constitutes a Bain path. To obtain a classification label, we select the class with the higher classification probability through an *argmax* operation (i.e., the label maximizing Eq. 2.3.2.1). These predictions are shown in Fig. 5.2.

The model is able to detect the bcc and fcc phases in the expected areas, while all prototypes not being fcc, bcc, or tetragonal are correctly labeled as "Other". We point out that only four structures – corresponding to points in the plot marked by the four stars – are included in the training set, while all other 3 717 structures are model (test) predictions. We can also observe that the model correctly predict the presence of a tetragonal phase between fcc (yellow band) and bcc (green band), even though no tetragonal structures from this region are included in the training set. This transition is smooth, only interrupted by small areas for which other, low-symmetry prototypes are assigned, but with high uncertainty, as quantified by the mutual information, cf. Fig. 5.2 c. We provide the classification

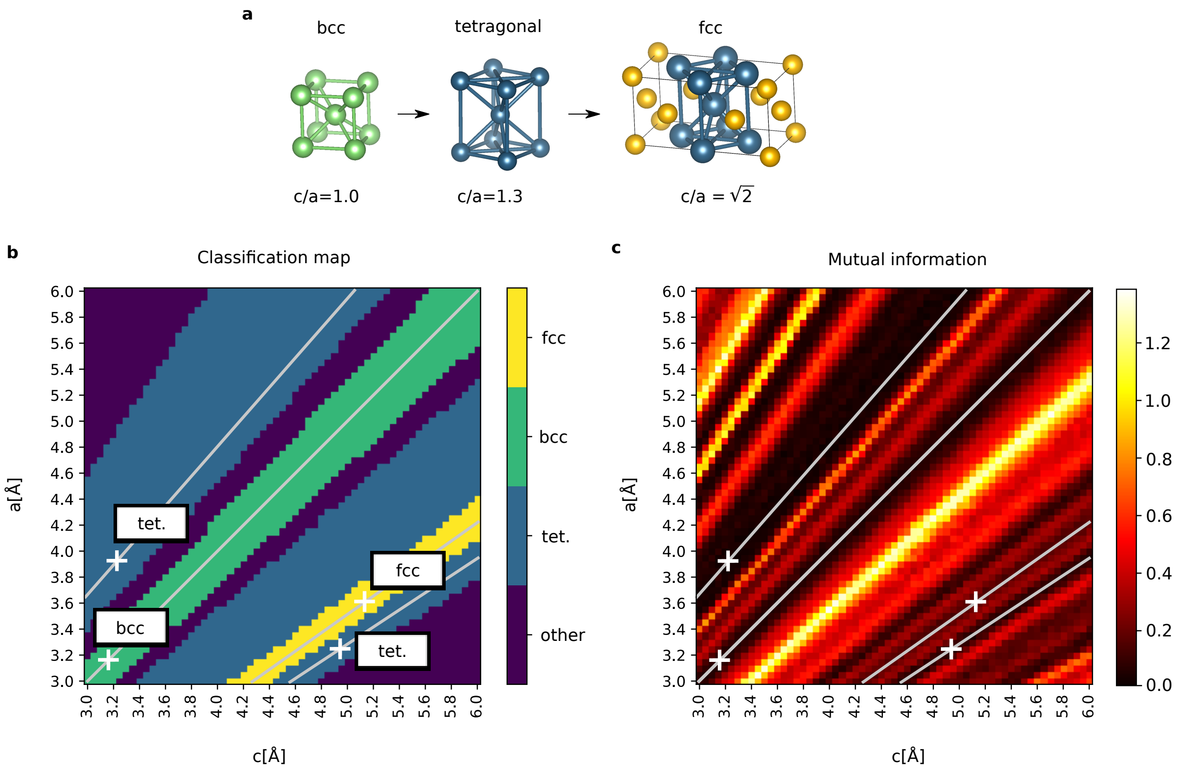

Figure 5.2: **a** Structures occurring in the Bain path, obtained by varying c/a; increasing c/a from 1.0 (bcc) leads to transitions to tetragonal (tet.) phases and finally to the fcc structure ($c/a = \sqrt{2}$) . **b** Classification (argmax predictions, left) and uncertainty (mutual information, right) for geometries in the range $c, a \in [3.0\,\text{Å}, 6.0\text{Å}]$. Geometries included in the training set are marked by stars in **b,c**. As we isotropically scale the structures, geometries with constant c/a are equivalent, which is indicated by solid lines.

probabilities of all assigned prototypes in Fig, C.1. In general, increased uncertainty appears at transitions between the assignments of different prototypes. We also note that there is a smooth transition for classification probabilities at the transition between prototypes (cf. Fig. C.1). These results represent a first indication that the network has learned physically meaningful representations. Surprisingly, for large or small c/a ratios, i.e., points far outside the training set, other (low-symmetry) phases appear such as base-centered orthorhombic molecular iodine or face-centered orthorhombic $\gamma-$Pu with small uncertainty. While it may be desirable to avoid overconfident predictions far away from the training set, the assignments could be actually physically justified given the similarities between tetragonal and orthorhombic lattices, the most evident being that all angles in both crystal systems are 90°. We note that the transition to these prototypes is encompassed by regions of high uncertainty also in this case in agreement with physical intuition.

5.4 When the model is forced to fail: analysis of ARISE out-of-sample predictions

To assess the physical content learned by the network, we investigate its predictions – and thus its generalization ability – on structures corresponding to prototypes not included in the training. This is of particular relevance if one wants to use predictions of ARISE – for applications such as screening of large databases, or create low-dimensional maps for a vast collection of materials [219].

Given an unknown structure, the network needs to decide – among the classes it has been trained on – which one is the most suitable. It will assign the most similar prototypes and quantify the similarity via classification probabilities, providing a ranking of candidate prototypes. The uncertainty in the assignment, as quantified by mutual information, measures the reliability of the prediction. Note that the task of assigning the most similar prototype(s) to a given structure among 108 possible classes (and quantifying the similarity) is a very complicated task even for trained materials scientists, in particular in case of complex periodic and possibly defective three-dimensional structures.

We consider three examples (cf. Fig. 5.3 left): fluorite and tungsten carbide (from AFLOW) where the correct labels are known, and one structure from the NOMAD encyclopedia (ID mp-684691 in materials project), for which the assigned space group is 1, i.e., no symmetry can be identified (via spglib). In all three cases there is no prototype in the dataset which represents a match for any of these structures. This is on purpose: the network will “fail” by construction since the correct class is not included in the possible classes the network knows (and needs to choose from). Analyzing how the network fails will give us insight on the physical content of the learned model. This test can also be viewed as discovering “unexpected similarities” across materials of different chemical composition and dimensionality.

Following the pipeline for single-crystal classification summarized in Fig. 4.1a-d, we compute classification probabilities and mutual information, yielding the assignments shown in Fig. 5.3 right. To rationalize the predictions shown in Fig. 5.3 from a physical standpoint, we inspect the substructures formed by the chemical species in both original and assigned

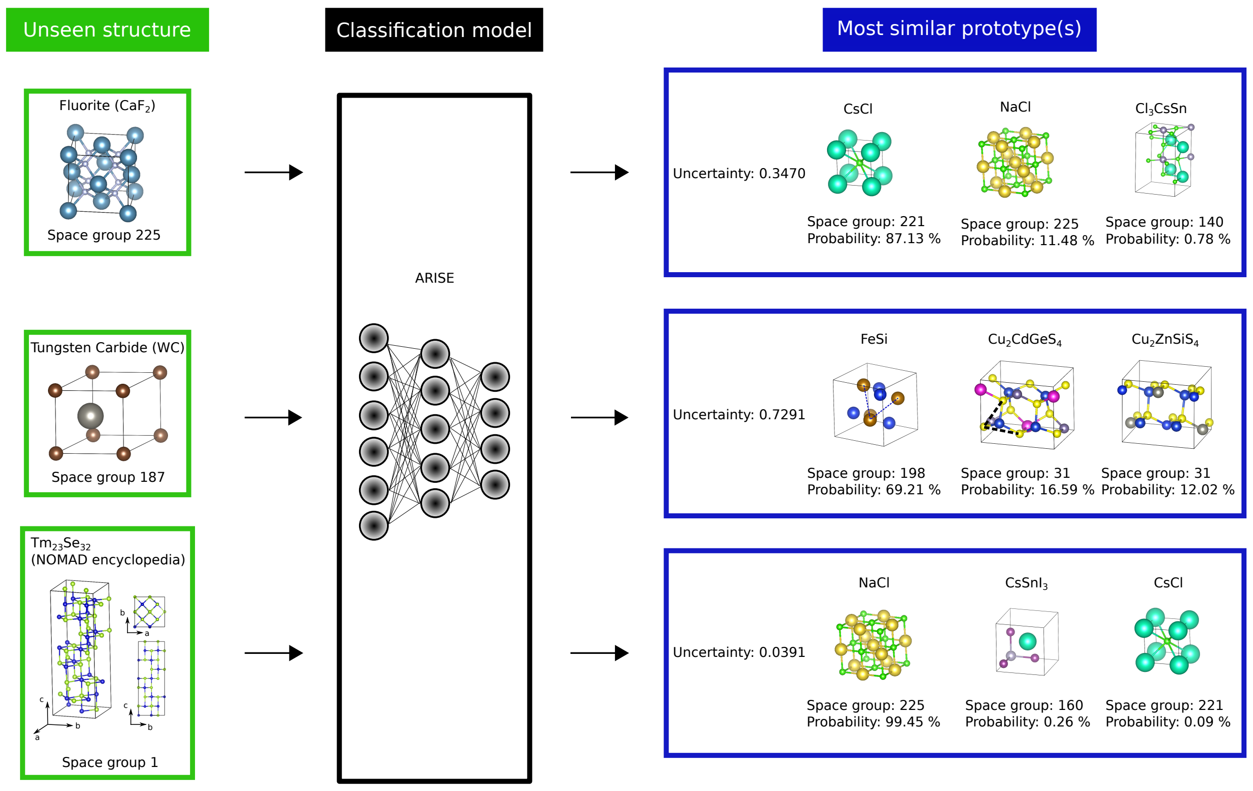

Figure 5.3: Three examples for assigning the most similar prototype(s) (right panel) to structures for which the corresponding structural class is not contained in the training set of ARISE (left panel). For each prediction, space group and classification probabilities of the top predictions are specified together with an uncertainty estimate (mutual information). The space groups are returned via spglib, where we choose the highest symmetry that is found for all combinations of precision parameters $(0.1, 0.01, 0.001, 0.0001)[\text{Å}]$ and angle tolerances $(1, 2, 3, 4, 5)[°]$.

structures. This is motivated by our choice of materials representation as averaged SOAP descriptor of substructures (cf. section 4.3.2). The two most similar prototypes to fluorite (CaF_2) are CsCl and NaCl, both consisting of two inter-penetrating lattices of the same type, two sc lattices for CsCl and two fcc lattices for NaCl. Fluorite contains both sc (F atoms) and fcc (Ca atoms) which is likely why CsCl and NaCl are assigned, together with a ternary halide tetragonal perovskite, also containing sc symmetry (via Cs and Sn atoms, respectively). For tungsten carbide (WC), W and C form two hexagonal lattices. In the unit cell of the most similar prototype, FeSi, 60° angles are formed within the substructures of each species (see dashed lines in the unit cell), thus justifying this classification. Furthermore, two quaternary chalcogenides appear as further candidates. This similarity – hard to assess by eye – originates by the presence of angles close to 60° for S atoms (yellow) for both Cu_2CdGeS_4 and Cu_2ZnSiS_4 (marked in the figure for Cu_2CdGeS_4). Also note that these two quaternary prototypes, Cu_2ZnSiS_4 and Cu_2CdGeS_4 are a result of substituting Ge and Si with isoelectric elements Zn and Cd, which implies that these structures are expected to be similar. This explains why they both appear as candidates for structures being similar to tungsten carbide.

Finally, for the compound $Tm_{23}Se_{32}$ from the NOMAD encyclopedia, the model identifies NaCl as the most similar prototype. Looking at the structure from different angles, especially from the top (cf. Fig. 5.3, left part), a similarity to cubic systems can be identified. The classification method robustness to missing atoms makes the apparent gaps in the side-view negligible, and thus rationalizes the NaCl assignment. Regarding the uncertainty quantification (via mutual information), increased uncertainties appear for fluorite and tungsten carbide, since besides the top prediction with more than 70% classification probability, other prototypes are possible candidates for the most similar prototype. For the NOMAD structure $Tm_{23}Se_{32}$, the network is quite confident, most likely because no other good candidates are presented among the binaries included in the 108 classes dataset.

These results show that the model – even when forced to fail by construction – returns (highly non-trivial) physically meaningful predictions. This makes ARISE particularly attractive for screening large and structurally diverse databases, in particular assessing structures for which no symmetry label can be obtained with any of the current state-of-the-art methods.

In addition to the analysis in Fig. 5.3, we report further examples for out-of-sample structures, whose ARISE-assignments can be justified in the same fashion. Each of these structures are taken from AFLOW. In the following list, we specify the input prototype name and the corresponding AFLOW URL (whose identifier specifies composition and symmetry, e.g., Boron nitride has the identifier AB_hP4_194_c_d, where AB denotes the stoichiometry, hP4 the Pearson symbol, 194 the space group, and c,d refer to the Wyckoff positions – see also section 12 of [101]). Moreover, we state the most likely prototype that is assigned by ARISE, as well as the associated mutual information (abbreviated as mut.inf.):

- Boron nitride (bulk, graphitic, http://aflowlib.org/CrystalDatabase/AB_hP4_194_c_d.html) classified as hexagonal graphite (probability 63.32%), mut.inf. 0.7278.
- Cementite (http://aflowlib.org/CrystalDatabase/AB3_oP16_62_c_cd.html) classified as MnP (orthorhombic) with probability 49.14%, mut.inf. 0.7176.
- $CuTi_3$ ($L6_0$ Srukturbericht, http://aflowlib.org/CrystalDatabase/AB3_tP4_123_a_ce.html) classified as $\alpha-$Pa (tetragonal) with probability 78.41%, mut.inf. 0.8539.
- Benzene (http://aflowlib.org/CrystalDatabase/AB_oP48_61_3c_3c.html) classified as nanotube (chiral indices (n,m)=(5,2)) with probability 68.48%, mut.inf. 0.6249.
- NbO (http://aflowlib.org/CrystalDatabase/AB_cP6_221_c_d.html), which is NaCl with 25% ordered vacancies on both the Na and Cl sites), classified as NaCl with probability 99.96%, mut.inf. 0.0027.
- Moissanite 4H SiC (http://aflowlib.org/CrystalDatabase/AB_hP8_186_ab_ab.html) classified as wurtzite with probability 99.74%, mut.inf. 0.0166
- K_2PtCl_6 (http://aflowlib.org/CrystalDatabase/A6B2C_cF36_225_e_c_a.html) classified as NaCl with probability 61.4%, mut.inf. 0.5402.

Chapter 6

Polycrystal characterization using ARISE

Up to this point, we have discussed only the analysis of single-crystal (mono-crystalline) structures, using ARISE. The only exception is the graphene structure analyzed in Fig. 5.1a (bottom row), which contains a grain boundary and is thus polycrystalline. To enable the local characterization of large, polycrystalline systems, we introduce *strided pattern matching* (SPM). This framework is explained in section 6.1, where section 6.1.1 describes the prediction workflow and section 6.1.2 discusses the most important parameters in the SPM framework. We consider a diverse set of applications: First, a mono-species benchmarking structure is analyzed in section 6.2, where both supervised and unsupervised protocols are established and tested in sections 6.2.1 and 6.2.2, respectively. The unsupervised study addresses an important topic in ML: model *explainability* [42–44]. Section 6.3 turns to a more realistic system of high technological relevance: Ni-based superalloys and the detection of precipitates in systems with random chemical ordering. A further realistic system is discussed in section 6.4, which in contrast to the previous systems is a result from atomistic simulations [98], specifically an evolutionary structure search based on the crystal-structure prediction method *USPEX* [220]. While sections 6.2-6.4 deal with simulated structures only, section 6.5 presents a study of experimental HAADF images of graphene via SPM (while the global analysis of comparable images is discussed in section 5.2). Moreover, the application of ARISE and SPM to experimental high-resolution transmission electron microscopy (HRTEM) images is discussed, specifically the classification of a quasicrystal, demonstrating the flexibility of our approach to deal with electron-microscopy data from diverse origins. Going up in spatial dimensionality, 3D AET data is analyzed in section 6.6, in both supervised (section 6.6.1) and unsupervised (section 6.6.2) fashion. In particular, section 6.6.2 establishes a protocol for the exploratory analysis of (atomic-resolution but) noisy structural data from experiment. Finally, adding a further dimension (the time), section 6.7 considers four-dimensional (4D), i.e., time-resolved AET data. This study supports that mutual information can be considered an AI-based order parameter.

6.1 The strided pattern matching framework

6.1.1 Prediction workflow

For slab-like systems (cf. Fig. 6.1 a), a box of predefined size is scanned in-plane across the whole crystal with a given stride; at each step, the atomic structure contained in the box is represented using a suitable descriptor (cf. Fig. 6.1 b-c), and classified (Fig. 6.1d), yielding a collection of classification probabilities (here: 108) with associated uncertainties. These are arranged in 2D maps (Fig. 6.1e). The classification probability maps indicate how much a given polycrystalline structure locally resembles a specific crystallographic prototype. The uncertainty maps quantify the statistics of the output probability distribution (cf. section 2.3.2). Increased uncertainty indicates that the corresponding local segment deviates from the prototypes known to the model. Thus, these regions are likely to contain defects such as grain boundaries, or more generally atomic arrangements different from the ones included in training. For bulk systems (Fig. 6.1f), the slab analysis depicted in Fig. 6.1a-e is repeated for multiple slices (Fig. 6.1g), resulting in 3D classification probability and uncertainty maps (Fig. 6.1h).

SPM extends common approaches such as labeling individual atoms with symmetry labels [114], as the striding allows to discover structural transitions within polycrystals in a smooth way. SPM can be applied to any kind of data providing atomic positions and chemical species. Results obtained via SPM are influenced by the quality of the classification model as well as box size and stride (see the next section for more details).

6.1.2 Parameter selection

Two parameters are most important for strided-pattern-matching analysis:

- Firstly, the stride defines the resolution and may be chosen arbitrarily small or large to increase or decrease the visualization of structural features. Note that the sliding allows us to discover smooth transitions, while the smoothness is determined by the step size. This way, boundary effects between neighbored local regions are reduced compared to the case of slightly overlapped or non-overlapping boxes (e.g., in the simple voxelization case). In particular, a small stride (e.g., 1 Å) mitigates boundary effects due to the discretization, which otherwise can influence the final classification and uncertainty maps. SPM is trivially parallel by construction, thus allowing the time-efficient characterization of large systems. Clearly, in a naive implementation, this procedure scales cubically with stride size. Practically, one may choose a large stride (in particular if the structure size would exceed computing capabilities) to obtain low-resolution classification maps, which may suffice to identify regions of interest. Then, one may zoom into these areas and increase the stride to obtain high-resolution classification maps revealing more intricate features.

- Secondly, the box size determines the locality, i.e., the amount of structure that is "averaged" to infer the crystallographic prototype being most similar to a given local region. If this parameter is chosen too large, possibly interesting local features may be smoothed out. We recommend to use box sizes larger than 10-12Å, as in these

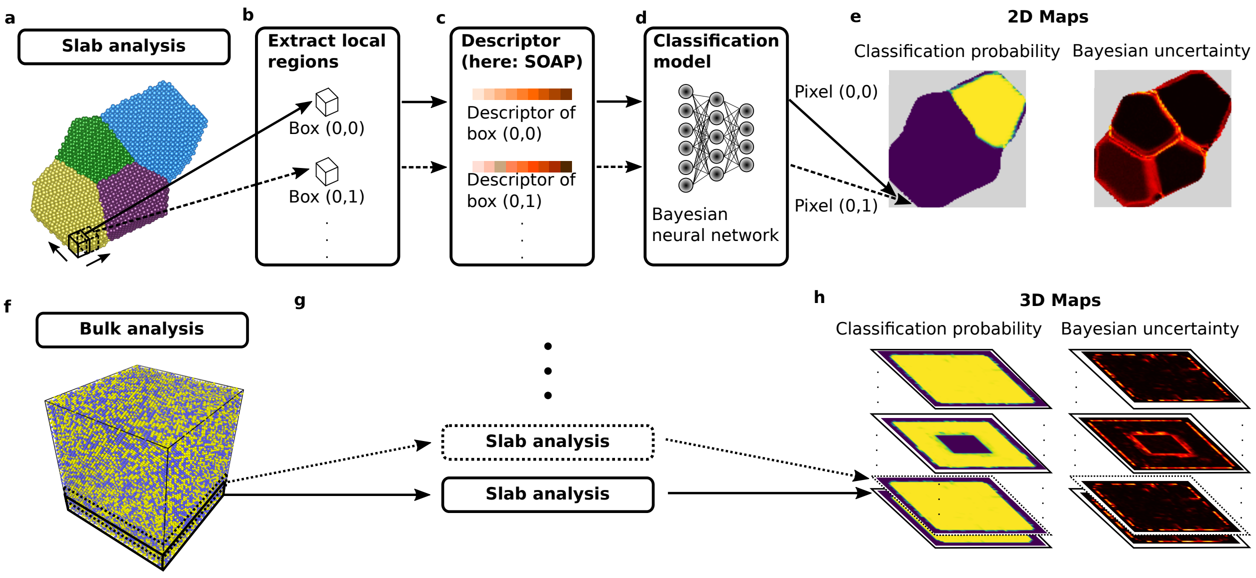

Figure 6.1: Schematic overview of the polycrystal characterization framework *strided pattern matching* for slab-like (**a-e**) and bulk systems (**f-h**).

cases, the number of contained atoms is typically within the range of the supercells the network is trained on (i.e., at least 100 atoms). The generalization ability to smaller structures depends on the prototype (cf. Fig. B.1), and in general, if a smaller box size is desired while using our model, the practical solution is to add smaller supercells in the training set and retrain the network.

Note that the shape of the local regions may be chosen to be different from boxes, e.g., spheres or any other shape that fits the application at hand. Moreover, we chose the grid in which the structure is strided to be cubic, while other discretizations are possible. Also note that a one-dimensional striding can be applied to rod-like systems such as carbon nanotubes.

As mentioned above (last sentence of the first bullet point), one can use the SPM algorithm to identify regions of interest. In the future, one may extend the SPM algorithm to conduct an adaptive multi-resolution analysis, i.e., the algorithm decides automatically where to zoom in, for instance, on the basis of the level of uncertainty from the previous levels of zoom.

6.2 Synthetic example I: mono-species polycrystal

6.2.1 Supervised analysis

First, the classification via SPM is performed for a slab-like synthetic polycrystal consisting of fcc, bcc, hcp, and diamond grains (cf. Fig. 6.2a, created via the open-source software Atomsk [221]). Since these symmetries cover more than 80% of the elemental solids found in nature [199], this structure is a solid testing ground for SPM. Due to the nature of the system, the SPM boxes near the grain boundaries will contain mixtures of different crystal structures. For the pristine structure, the results are shown in Fig. 6.2 b and c: We choose a 1 Å stride

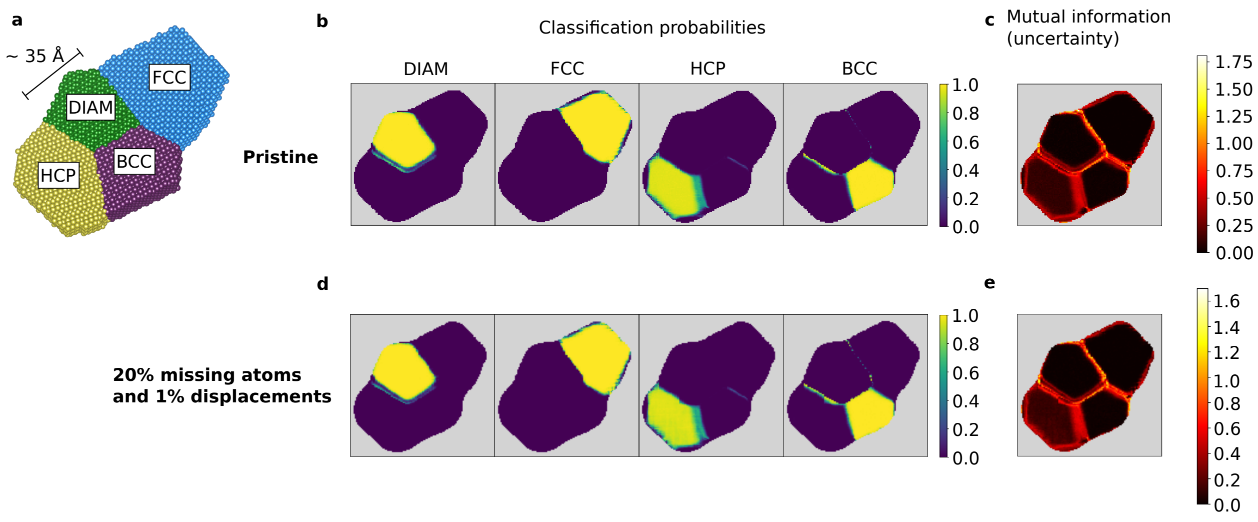

Figure 6.2: Supervised analysis of mono-species polycrystal consisting of four grains with different symmetries (**a**), for the pristine (**b-c**) and defective (**d-e**) case. **b,d** Classification probabilities of expected prototypes. **c,e** Mutual information map for uncertainty quantification.

and a box size equal to the slab thickness (16 Å). In total, this yields 7 968 local boxes. The network assigns high classification probability to the correct prototypes. Uncertainty is low within the grains, increasing at grain boundaries and crystal outer borders in line with physical intuition. The result remains virtually unchanged when introducing atomic displacements (up to 1% of the nearest-neighbor distance) while concurrently removing 20% of the atoms (cf. Fig. 6.2d, e). The highest classification probabilities (after from the top four shown in Fig. 6.2b, d) are shown in Fig. D.2; a discussion on the stride can be found in Fig. D.3.

6.2.2 Explainability of the black-box model via unsupervised learning

Going beyond classification, we show how unsupervised learning can be used to access structural similarity information embedded in ARISE's internal representations, and use it for the characterization of crystal systems. Moreover, this analysis allows to explain the internals of the black-box neural-network model. We consider the pristine mono-species polycrystal shown in Fig. 6.2a and collect ARISE's representations of the overall 7 968 local boxes. Next, we employ HDBSCAN [177, 178] to identify clusters in the high-dimensional representation space. HDBSCAN estimates the density underlying a given data set and then constructs a hierarchy of clusters, from which the final clustering can be obtained via an intuitive and tunable parameter (cf. section 3.1.3). The obtained clusters correspond to the four crystalline grains in the structure (Fig. 6.3a). Points identified as outliers (marked in orange) coincide with grain-boundary and outer-border regions. Next, the high-dimensional manifold of the NN representations is projected in 2D via UMAP [185]. UMAP models the manifold underlying a given dataset and then finds a low-dimensional projection that can capture both global and local distances of the original high-dimensional data (cf. section 3.2.1). This returns a structure-similarity map (Fig. 6.3b), which allows to visually investigate similarities among structures: points (structures) close to each other in this map are considered to be similar by the algorithm. Structures belonging to the same cluster are in close proximity to each other, and clearly separated from other clusters. Conversely, outlier points are split across different regions of the map. This is physically meaningful: outliers are not a cohesive cluster of similar structures, but rather comprise different types of grain boundaries (i.e., fcc to bcc transitions or fcc to diamond etc., cf. Fig. D.4). In this synthetic setting, we can also use the classification prediction to further verify the unsupervised analysis: the results obtained via unsupervised learning indeed match ARISE's predictions (cf. Fig. 6.3b - Fig 6.3c). Moreover, an analysis of the mutual information (Fig. 6.3d) reveals that points at the core of the clusters are associated with low uncertainty, while points closer to the boundaries show increased uncertainty. Similar results are obtained for the other layers (cf. Fig. D.1).

The analysis conducted in Fig. 6.3 qualifies as a *post hoc explanation* [222] of the trained model – a term on which we will elaborate in the following. As we discussed on multiple occasions in this thesis (for instance, section 2.2.3), generalization ability is the key task in ML. This does not only mean that a ML model performs well on a test set sampled from a distribution similar to the one of the training set, but actually provides meaningful predictions for significantly different data. To assess what is meaningful or not, one needs to understand what the model has learned. This task is directly connected to *interpretability*, a field of ML whose definition is being debated at a fundamental level [42–44]. Two categories of interpretability are typically considered [222]: *transparency* addresses the functioning of the internal mechanisms of a ML model. Deep-learning models lack transparency due to their black-box nature, a consequence of the information being distributed across millions of parameters. *Post hoc explanations* address the information one can extract from a (black-box) ML model, while the detailed internal mechanisms typically remain hidden. For instance, one can relate the change of model input and output, revealing which parts of the input are more important for certain predictions. In crystal-structure identification, *attentive response maps* have been employed in this fashion [26], determining which parts of a diffraction fingerprint influence the structural-class prediction most. The diffraction fingerprint that represents a given crystal is essentially a 2D image with pronounced peaks. It is shown in [26] that the trained deep-learning model employs these peaks to arrive at a classification, in a similar way as a trained materials scientist would do. In this work, the input lacks comparable interpretability, since we employ the SOAP descriptor whose components are essentially basis set expansion coefficients. Still, to understand which physical insights the model has learned (or if any at all) we can change the input. For instance, we have continuously deformed the input (via studying structural transformation paths in section 5.3) or investigated the predictions for out-of-sample structures (cf. section 5.4). In this section, we employed unsupervised learning to study the internal representations that have been learned by the neural network model. Specifically, we employed clustering (HDBSCAN) and related the obtained grouping of points to structural characteristics in real space. Moreover, we used manifold learning (UMAP) to obtain a low-dimensional, easily interpretable structural-similarity map. We find that despite the original high-dimensionality, the compressed representation reveals meaningful clusters that are again related to real-space patterns. In section 6.6.2, we will provide an additional *post hoc* explanation study that also serves as a protocol for exploratory analysis of crystal structures.

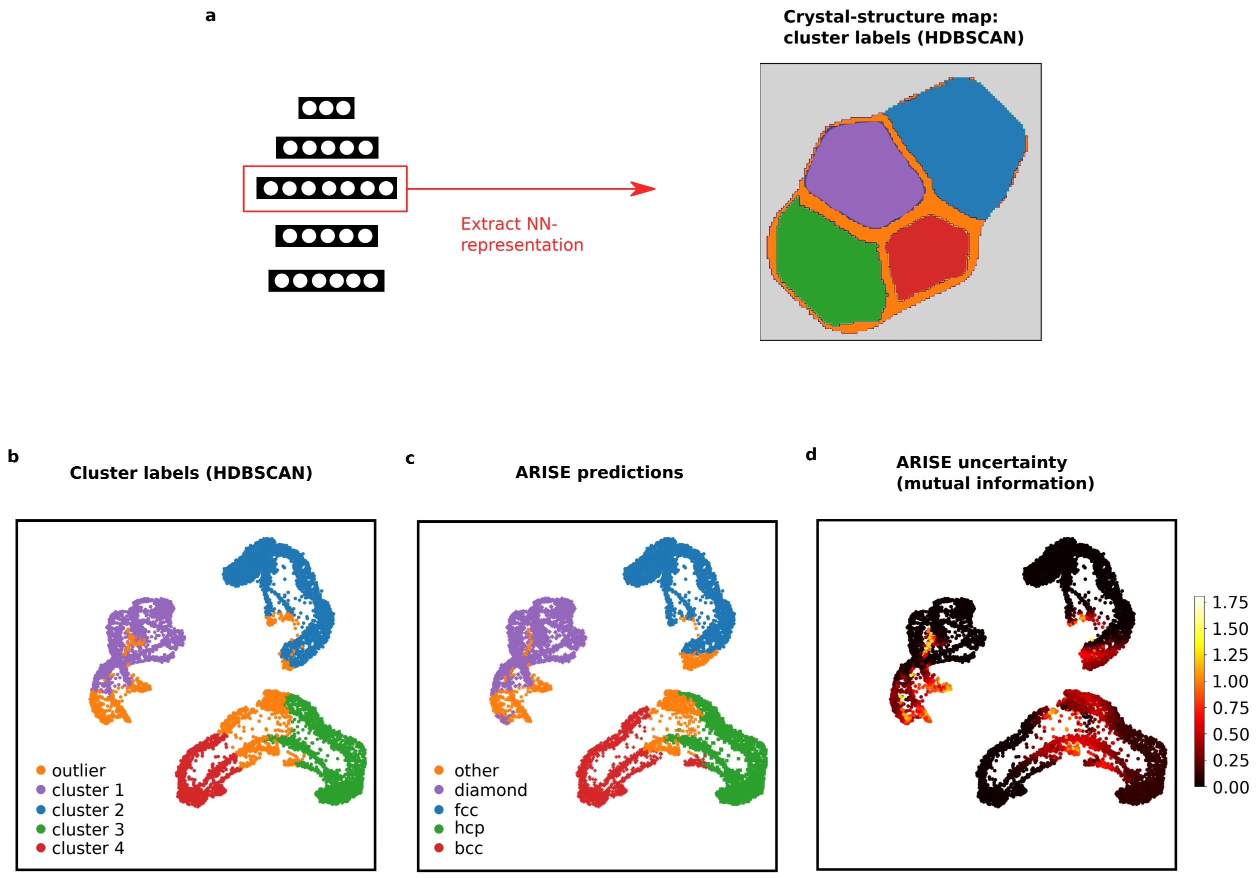

Figure 6.3: **a-d** Unsupervised analysis of internal neural-network representations. **a** The neural-network (NN) representations are extracted for each local segment in Fig. 6.2 **a** (obtained via SPM for the pristine structure). Clustering (via HDBSCAN) is applied to this high-dimensional internal space; the polycrystal is marked according to the resulting clusters (see legend in **b** for the color assignments). **b-d** Two-dimensional projection (via UMAP) of neural-network representations colored by cluster label, ARISE predicted class, and mutual information, respectively. In **b**, all points for which HDBSCAN does not assign a cluster are labeled as “outlier”. In **c**, all points that are not classified as fcc, diamond, hcp or bcc are labeled as “other”. Note that while the distances between points are meaningful, the axes merely serve as a bounding window and are not interpretable – a situation typically encountered in non-linear methods such as UMAP (cf. section 6 in [185]).

6.3 Synthetic example II: Ni-based superalloy

We now move to a more realistic system: a model structure for Ni-based superalloys [201] (c.f Fig. 6.4a). Ni-based superalloys are used in aircraft engines due to their large mechan-

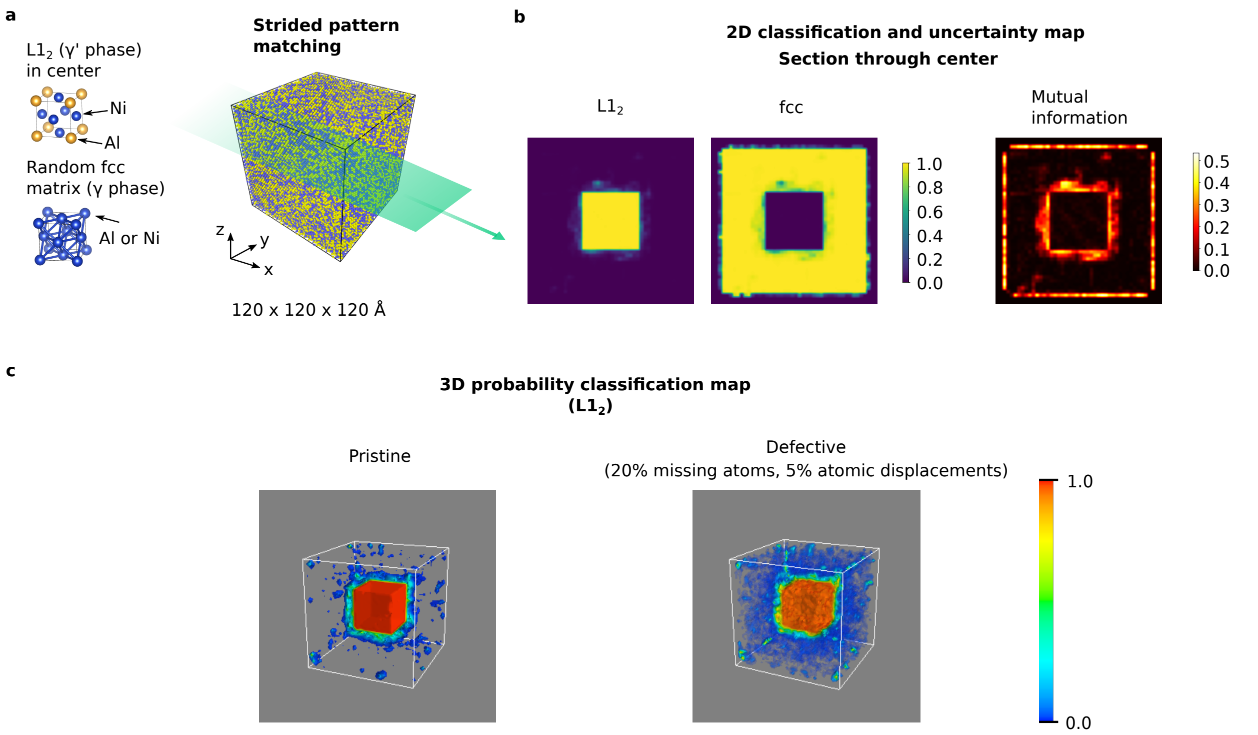

Figure 6.4: Precipitate detection in Ni-based superalloys. **a** Binary model system (right) and depiction of the two appearing phases (left). **b** Classification probabilities of expected prototypes and mutual information for a slice through the center of the structure. **c** 3D-resolved detection of the precipitate via the $L1_2$ classification probability for the pristine (left) and highly defective case (right), for which 20% of the atoms are removed and randomly displaced (up to 5% of the nearest-neighbor distance).

ical strength at high temperatures, which derives from ordered $L1_2$ precipitates (γ' phase) embedded in a fcc matrix (γ phase). We generate an atomic structure consisting of a fcc matrix in which Al and Ni atoms are randomly distributed. In the center, however, the arrangement of Al and Ni atoms is no longer random, but it is ordered such that the $L1_2$ phase is created (c.f Fig. 6.4a). The cubic shape of this precipitate is in accordance with experimental observations [223]. The resulting structure comprises 132 127 atoms over a cube of 120 Å length. Compared to Fig. 6.2, we choose the same box size (16 Å) but reduce the stride to 3 Å, since this system is much larger and we want to demonstrate that in these situations, smaller strides (that reduce computational cost) still yield reasonable results. As shown via a section through the center in Fig. 6.4b, fcc is correctly assigned to the matrix, and the precipitate is also detected. The uncertainty is increased at the boundary between random matrix and precipitate, as well as at the outer borders. Fig. 6.4c illustrates the $L1_2$ classification probability in a 3D plot. The precipitate is detected in both pristine and highly defective structures. This analysis demonstrates that ARISE can distinguish between chemically ordered and chemically disordered structures, a feature that will be exploited for the analysis of experimental AET data in sections 6.6 and 6.7.

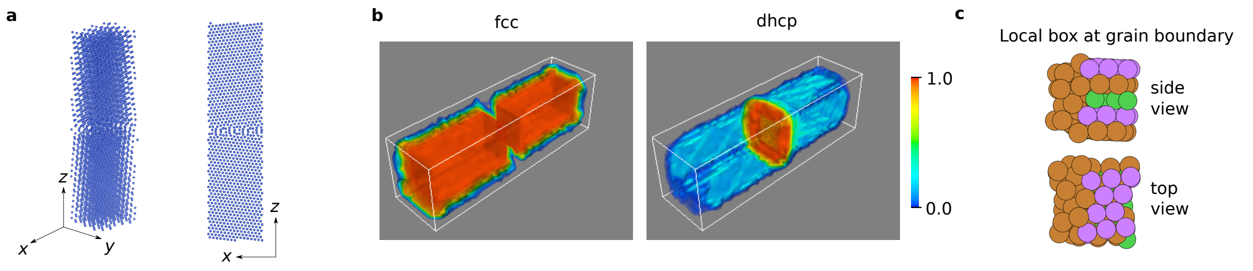

Figure 6.5: **a** Lowest-energy grain boundary structure (Cu, fcc) predicted from an evolutionary search (based on the crystal-structure prediction method *USPEX* [220]). The geometry data is taken from the original reference [98]. The so-called Pearl pattern appears at the grain boundary, which is also observed in experiment [98]. **b** ARISE analysis (via SPM), correctly identifying fcc (ABC close-packing) in the grains, while detecting double hexagonal close-packed (dhcp, ABAC) at the grain boundary. **c** Exemplary analysis of a local box at the grain boundary, illustrating a change in stacking and increased distortions, which motivates the assignment of dhcp (which contains 50 % of both fcc and hcp close-packings).

6.4 Synthetic example III: grain-boundary system obtained via evolutionary structure search

In this section we study the structure shown in Fig. 6.5a, which is the lowest-energy structure obtained from an evolutionary structure search [98] (based on the crystal-structure prediction method *USPEX* [220]). Notably, the structural patterns at the grain boundary (termed "Pearl") are also observed in STEM experiments. This study supplements the systems investigated in section 6.2 and 6.3, which unlike the system in Fig. 6.5a are not created using atomistic simulations.

ARISE (via SPM) correctly identifies the fcc symmetry within the grains (cf. Fig. 6.5b) while assigning double hexagonal close-packed (dhcp) symmetry at the grain boundary. The local boxes at the grain boundary contain partial fcc structures while changes in stacking and distortions decrease their symmetry (cf. Fig. 6.5c). Also the dhcp phase (ABAC close-packing) contains fcc (ABC) and a lower-symmetry packing (hcp, AB), thus justifying the assignment. A box size of 10 Å and a stride of 2 Å suffice to characterize this grain-boundary structure.

Note that ARISE correctly identifies even the $\alpha-$Sm-type stacking (ABCBCACAB). No other fully automatic approach offers a comparable sensitivity.

6.5 Application to STEM and HRTEM experimental images

In section 5.2 we have analyzed HAADF images on a global scale via ARISE. Using SPM, a local analysis can be performed, specifically to zoom into a given structure and locate substructural features. This is particularly useful for polycrystalline and/or larger systems (e.g., more than 1 000 atoms). As illustrative example, we consider the structure in Fig. 6.6a. We choose a stride of 4 (in units of pixels since atoms are reconstructed from images, while for typical graphene bond lengths of 1.42 Å the relation 1 Å $\approx$ 8.5 can be inferred). Moreover, we select a box size of 100 pixels ($\approx$ 12 Å). The mutual information shown in Fig. 6.6b clearly reveals the presence of a grain boundary. In Fig. 6.6c , the classification probabilities of graphene and MnS_2 (the dominant prototypes) are presented, the latter being assigned at the grain boundary. This assignment can be traced back to pentagon-like patterns appearing near the grain boundary (as highlighted in Fig. 6.6a, right), a pattern similar to the one being formed by Mn and S atoms in MnS_2 (cf. Fig. 6.6d).

This analysis together with the global study presented in Fig. 5.1 provides a blueprint for the study of STEM and in general atomic-resolution electron microscopy images, adding to the growing body of research addressing this field [224]: Given an atomic-resolution image, reconstruct the atomic positions and then apply ARISE for further analysis (assigning the underlying lattice symmetry, detecting regions of interest such as grain boundaries). Note that images investigated in Fig. 5.1 and Fig. 6.6 contain a significant amount of noise, hindering the application of standard reconstruction methods that typically rely on the fitting of 2D Gaussians (e.g., *atomap* [225]). Fortunately, experimentalists routinely achieve much better resolution, making the application of classical (i.e., non-data-driven) methodologies possible. In particular, AtomNet is trained on particular systems only and generalization to other symmetries is not guaranteed (while models that can detect Si defects or cubic symmetry are available). Notably, even if AtomNet or other methods provide incomplete information, ARISE's robustness can be employed to still obtain reasonable results.

To support the discussion in the previous paragraph, we challenge the established procedure for the local analysis of 2D images with data from a completely different resource. We investigate a HRTEM image of a quasicrystalline structure [226, 227], cf. Fig 6.7a. The bright spots are ordered aperiodically, making it a very hard task to identify the underlying order by eye. Via the established procedure (using a box size of 100 and a stride of 10 pixels in the 1000×1000 pixel image), MnS_2 is predicted as the most similar prototype (cf. Fig. 6.7b). MnS_2 contains pentagon patterns (cf. Fig. 6.6d) which can also be seen in the quasicrystal (cf. zoom in Fig. 6.7a). Besides the confirmation of the previous analysis in Fig. 6.6, this result suggests that ARISE and SPM are novel and promising tools for the classification of quasicrystalline order in automatic fashion – a promising yet under-explored area.

6.6 Application to 3D atomic electron tomography data

While HAADF (or HRTEM) images are a valuable experimental resource, they only provide 2D projections. 3D structural and chemical information can however be obtained from atomic electron tomography (AET) with atomic resolution achieved in recent experiments [109, 228–231]. Notably, this technique provides 3D atomic positions labeled by chemical species, to which ARISE and SPM can be readily applied. While time-resolved experiments [232] and extensions to other systems such as 2D materials are reported [233], metallic nanoparticles have been the main experimental focus so far, specifically FePt systems due

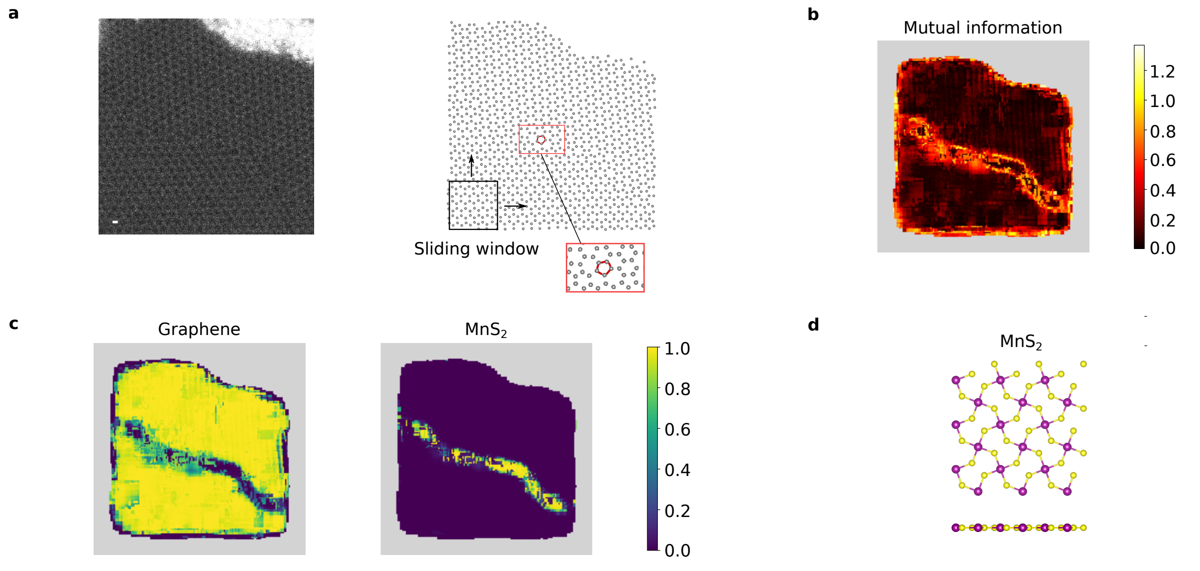

Figure 6.6: Analysis of HAADF image (from https://github.com/pycroscopy/AICrystallographer/tree/master/AtomNet) via ARISE and SPM. **a** HAADF image and reconstructed atomic positions (analogous to Fig. 5.1 via AtomNet [81]) of a larger sample. Pentagons can be spotted near the grain boundary (see inset). The white scale bar in the bottom left of the image corresponds to the typical graphene bond length (1.42 Å). **b-c** Local analysis: graphene is the dominant structure (**c**). Different prototypes (MnS_2, cf. **d**) are only assigned - and with high uncertainty (mutual information, **b**) - at the grain boundary. **d** MnS_2 prototype.

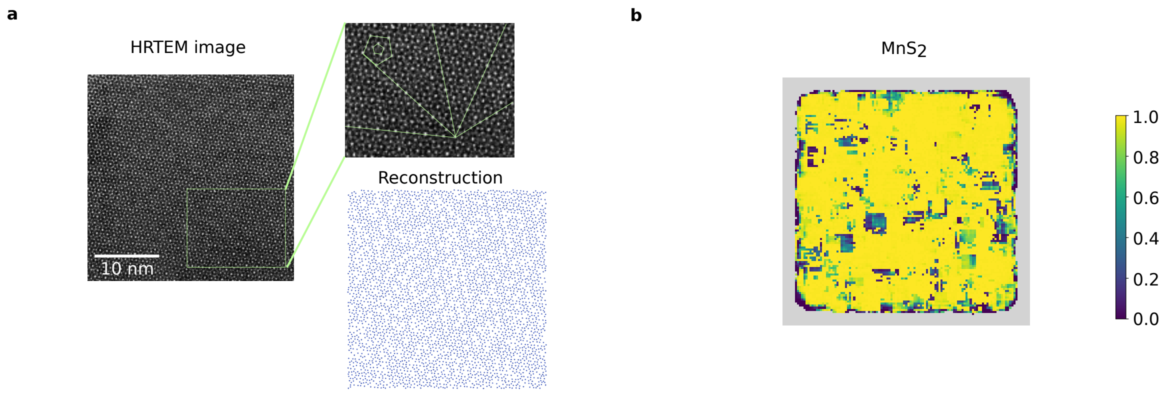

Figure 6.7: Analysis of HRTEM image via ARISE and SPM. **a** HTREM image of quasicrystalline structure (icosahedral Al-Cu-Fe, adapted from the original reference [227], i.e., cropped and rescaled to a 1000×1000 pixel image using standard settings in the GIMP Image editor). While there is an underlying order, the structure is aperiodic (i.e., no translational symmetry is present). As visualized in the zoom, the bright spots align with five-fold symmetry axes and pentagons of different size appear. Based on the reconstruction via AtomNet (bottom right), ARISE (via SPM) identifies MnS_2 as the dominating prototype (**b**), which similarly to the input structure contains pentagon patterns (cf. Fig. 6.6d).

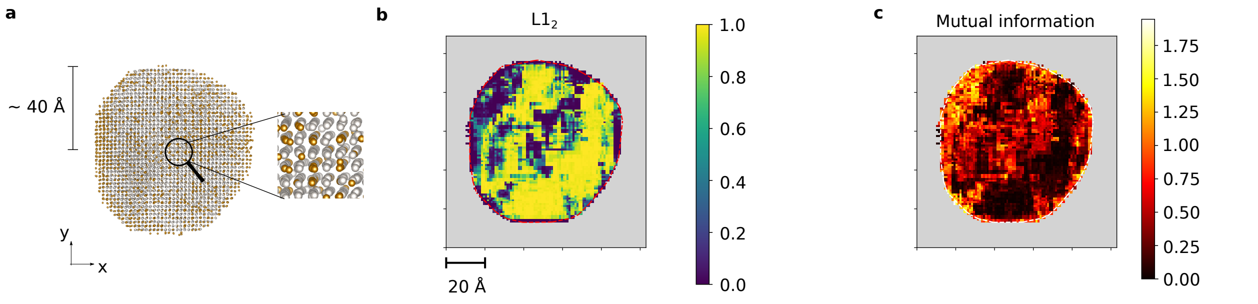

Figure 6.8: Supervised analysis of AET data. **a** Side view of FePt nanoparticle (∼23k atoms), with atomic positions and chemical species from AET data [237]. **b** L1$_2$ is the most dominant phase, which is exemplarily demonstrated via the classification probability of the corresponding prototype for the most central slice obtained using SPM. **c** The mutual information is low within regions dominated by L1$_2$, indicating increased crystallinity. These results are in accordance with a previous study [237]. Particle boundaries are indicated with red (**b**) and white (**c**) dashed lines.

to their promises for biomedicine and magnetic data storage [234]. Notably, a recent study employed AET to analyze amorphous solids [235, 236], demonstrating the flexibility of this experimental approach.

6.6.1 Supervised analysis

First, a FePt nanoparticle [237] is classified using SPM. ARISE's robustness is critical for this application, since the structural information provided by AET experiments are based on reconstruction algorithms that causes visible distortions (cf. Fig. 6.8a). We choose a stride of 1 Å and box size of 12 Å (see also the last paragraph in section 6.6.2 for a motivation of this selection). SPM primarily detects L1$_2$ (cf. Fig. 6.8b) but also L1$_0$, and fcc phases (cf. Fig. D.5). This is in line with physical expectations: annealing leads to structural transitions from chemically disordered to ordered fcc (A1 to L1$_2$) or to the tetragonal L1$_0$ phase [234, 237]. Besides the expected prototypes, ARISE also finds regions similar to tetragonally distorted, mono-species fcc (cf. Fig. D.5), which is meaningful given the presence of fcc and the tetragonal phase L1$_0$.

6.6.2 Exploratory analysis via unsupervised learning

To go beyond the information provided by classification and discover hidden patterns and trends in AET data, we conduct an exploratory analysis using unsupervised learning on ARISE's internal representations. While the procedure is similar to the one presented in Fig. 6.3, here the analysis is truly exploratory (no ground truth is known), and data comes from experiment. First, all SPM boxes classified as L1$_0$ are extracted, this choice motivated by the physical relevance of this phase, in particular due to its magnetic properties [234]. This reduces the number of data points (boxes) from 43 679 to 5 359 – a significant filtering step for which the automatic nature of ARISE is essential. In the representation space of the first hidden layer, HDBSCAN identifies seven clusters (and the outliers). To interpret the cluster

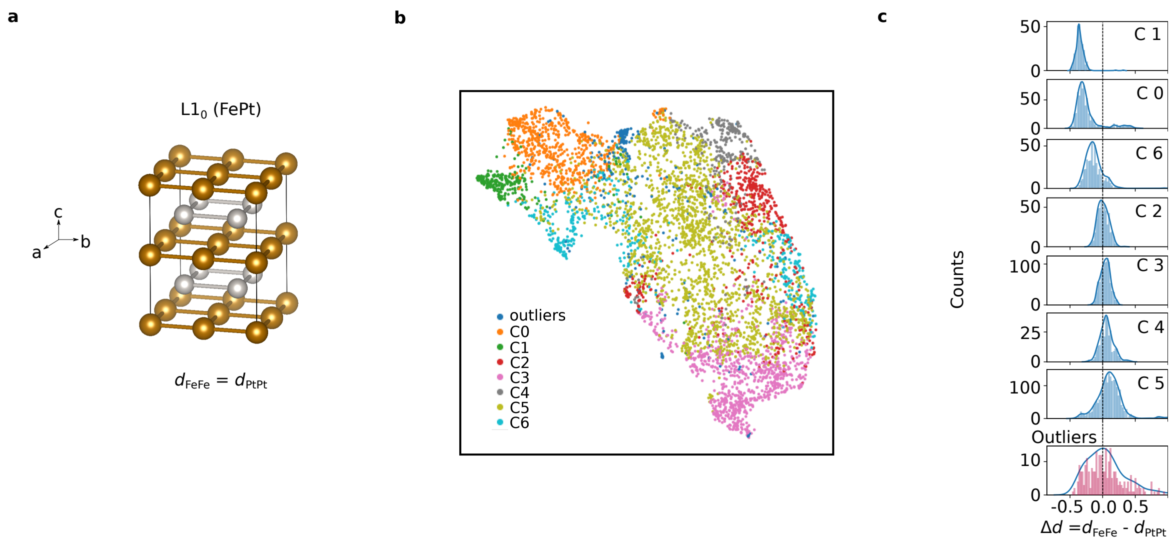

Figure 6.9: Unsupervised, exploratory analysis of AET data from [237]. **a** Illustration of $L1_0$ phase. All local regions that are assigned this technologically relevant prototype are extracted for further, unsupervised analysis. **b** Two-dimensional projection of neural-network representations (first hidden layer) via UMAP for regions classified as $L1_0$ by ARISE. **c** The distribution of the difference between the nearest-neighbor distances d_{FeFe} and d_{PtPt} (highlighted by bonds in **a**) is shown for each cluster.

assignments, we analyze geometrical characteristics of atomic structures (i.e., the local boxes) assigned to the different clusters. Specifically, we consider the nearest-neighbor distances between Fe and Pt atoms, d_{FeFe} and d_{PtPt}, respectively (cf. section 4.3.1 for the definition). For an ideal tetragonal structure, the difference $\Delta d = d_{\mathrm{FeFe}} - d_{\mathrm{PtPt}}$ is zero (cf. Fig. 6.9 a); a deviation from this value thus quantifies the level of distortion. Looking at the histograms of the (signed) quantity Δd shown in Fig. 6.9c for each cluster, one can observe that each distribution is peaked; moreover, the distribution centers vary from negative to positive Δd values across different clusters. The distribution of the outliers is shown for comparison: the Δd distribution is very broad, since outlier points are not a meaningful cluster. While overlap exists, the clusters correspond to subgroups of structures, each distorted in a different way, as quantified by Δd. Thus, we discovered a clear trend via the cluster assignment that correlates with the level of distortion. The cluster separation can be visualized in 2D via UMAP (cf. Fig. 6.9b). Notably, the clusters do not overlap, even in this highly compressed representation (from 256 to 2 dimensions). Some of the clusters may also contain further sub-distributions, which seems apparent, for instance, from the Δd distribution of cluster 6. The regions corresponding to the clusters could be hinting at a specific growth mechanism of the $L1_0$ phase during annealing, although further investigations are necessary to support this claim. The present analysis provides a protocol for the machine-learning driven exploration of structural data: supervised learning is employed to filter out a class of interest (which is not a necessary step, cf. Fig. 6.3), then unsupervised learning is applied to the neural-network representations, revealing regions sharing physically meaningful geometrical characteristics.

We want to emphasize that the analysis in this section also provides a *post hoc* explanation of the trained model, similar to the study presented in section 6.2.2. In this case, however, we focus on one particular phase (of high technological relevance) and then use a geometrical property to justify the clusters that are found in the internal neural-network space. In section 6.2.2, the real-space symmetry within each grain is employed to explain the clusters, while no filtering is performed.

In the following, we discuss the choice of SPM parameters in the current section as well as section 6.6.1. For Fig. 6.8, we choose a stride of 1 Å and box size of 12 Å. For the unsupervised analysis in Fig. 6.9, we reduce the stride to 2Å, to avoid an overcrowded 2D map. The box size of 16 Å which allowed to distinguish chemically disordered fcc from ordered $L1_2$ (cf. Fig. 6.4) yields comparable results (see Fig. D.5), while finding less $L1_0$ symmetry and more fcc since a larger amount of structure is averaged. Due to $L1_0$ showing special magnetic properties, we are interested in having a larger pool of candidate regions, which is why we choose a box size of 12 Å (corresponding to the smallest value such that the average number of atoms in each box is greater than 100).

6.6.3 Important parameters for the unsupervised explanatory and exploratory analysis protocols

In this section, we discuss the most important parameters that have been employed to perform the explanatory and exploratory analysis discussed in section 6.2.2 and 6.6.2.

As discussed in section 3.1.3, HDBSCAN [177, 178] is a density-based, hierarchical clustering algorithm. The final *flat clustering* is derived from a hierarchy of clusters. The most influential parameter is the minimum cluster size that determines the minimum number of data points a cluster has to contain – otherwise it will be considered an outlier, i.e., not being part of any cluster. Practically, one can test a range of values for the minimum cluster size, in particular very small, intermediate and large ones – for instance, for the results on the synthetic polycrystal in 6.2, we test the values $\{25, 50, 100, 250, 500, 1\,000\}$. In line with intuition, the number of clusters grows (shrinks) for smaller (larger) values of minimum cluster size. A coherent picture with 4 clusters and clear boundaries (as indicated by the outliers) arises for minimum cluster size values of around 500, for which we report the results in Fig. 6.3a-d and Fig. D.1. For the nanoparticle data discussed in Fig. 6.9, we observe that most of the points are considered outliers since the data contains substantially more distortions. To address this, we use the soft clustering feature of HDBSCAN, which allows to calculate a vector for each data point, whose ith component quantifies the probability that the given data point is member of cluster i. Then, we can infer a cluster assignment for points that would normally be considered outliers, by selecting for each point the cluster whose membership probability is maximal (while considering a point an outlier if all probabilities are below a certain threshold for which we choose 10 %). For the minimum cluster size, we find that for values below 10 the number of clusters quickly grows while shrinking for larger values. We report the results for a minimum cluster size of 10 in Fig. 6.9.

To visualize the clustering results, we use the manifold-learning technique UMAP [185] (cf. section 3.2.1). This method uses techniques from Riemannian geometry and algebraic topology to capture both the global and local structure of a manifold that underlies a given dataset. One of the most important parameters is the number of neighbors that will be considered to construct a topological representation of the data, where a small value takes only the local structure into account, while a large value considers the global relations between data points. We choose values of 500 for Fig. 6.3 and 50 for 6.9b, above which the 2D embeddings do not change significantly. For the minimum distance parameter (which is explained in the last paragraph in section 3.2.1), we employ the standard setting of 0.1 for 6.9b. For Fig. 6.3, we test several settings in Fig. D.4. For all tested values, the main characteristics can be resolved (in particular, four clusters corresponding to the four crystalline grains). In line with intuition, for smaller minimum distances, the points are tightly clustered, while for larger values, the points appear more spread, allowing for an improved visualization of transitions between the grains. We report the result for a minimum distance of 0.9 in Fig. 6.3.

6.7 Application to 4D atomic electron tomography data

Finally, we apply ARISE to time-resolved (4D) AET data. Specifically, a nanoparticle measured for three different annealing times is investigated [232]. The mutual information as obtained via SPM is shown in Fig. 6.10 for five central slices. We use a stride of 1 Å and a box size of 12 Å (i.e., the same parameters that have been chosen for Fig. 6.8 and 6.9). In regions between outer shell and inner core, the mutual information clearly decreases for larger annealing times, indicating that crystalline order increases inside the nanoparticle. Fig. D.6 supports the visual impression of Fig. 6.10 by a quantitative analysis. This study confirms that the predictive uncertainty of ARISE, as quantified by the mutual information, directly correlates with crystalline order. The mutual information can be therefore considered an AI-based order parameter, which we anticipate to be useful in future nucleation dynamics studies.

Notably, it is an important and non-trivial result that an information-theory concept such as mutual information applied to a complex deep-learning model correlates with physical intuition. Besides the study in Fig. 6.10, correlation of mutual information with the level of defectiveness has also been observed for multiple other scenarios, e.g., for bulk systems at grain boundaries (cf. Fig. 6.2), as well as 2D materials for both global (cf. Fig. 5.1) and local studies (cf. Fig. 6.6).

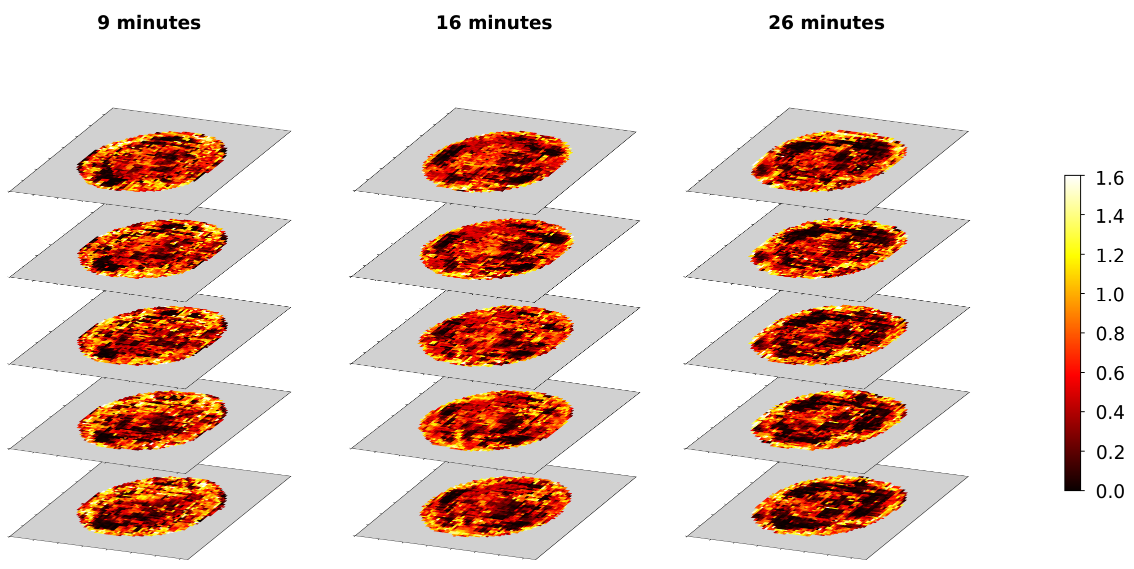

Figure 6.10: Five central slices (mutual information, obtained via SPM) for three different annealing times (data from 4D AET experiment [232]). The mutual information decreases for larger annealing time, in particular in regions between outer shell and inner core. This confirms that the predictive uncertainty of ARISE (as quantified by mutual information) correlates with crystalline order.

Chapter 7

Summary

Artificial intelligence (AI) is impacting various scientific fields including materials science. AI tools such as artificial neural networks (ANNs) are a major driving force of an emerging research paradigm: data-centric materials science [25]. An ultimate goal is to find hidden patterns and trends in data, leveraging the unique pattern-recognition abilities of AI methods such as ANNs. An important step in the characterization of a material is the analysis of its crystal structure, which is essential to predict physical properties [238, 239]. Quantifying the degree of symmetry or rather structural order that is contained in a given atomic structure is addressed in crystal-structure classification [114]. In general, given atomic coordinates and chemical composition, one is interested in finding the most similar structure within a list of given known systems. While periodicity is often assumed in theoretical studies, realistic materials are usually polycrystalline, exhibiting structural defects whose complete physical modeling is typically infeasible. For instance, grain boundaries induce peculiar mechanical properties in industrial steel [96]. To improve understanding of single- and polycrystalline materials, one can make use of vast and growing amount of structural information that is becoming available from both simulations and experiments: Large computational databases contain millions of structures alongside calculated properties, e.g., NOMAD (NOvel MAterials Discovery) [23, 24], AFLOW [101, 102], OQMD (Open Quantum Materials Database) [103], Materials project [104], and others [105, 106, 205]). Electron-microscopy experiments allow to study real materials at the atomic scale [108, 109, 228–233, 235]. To analyze this data, automatic methods for crystal-structure recognition are needed. The available structural data is often incomplete and noisy (due to physical defects or post-processing artifacts), thus requiring a robust crystal-recognition method. Given the structural diversity of materials space, flexibility, i.e., the ability to treat various systems, is a further key requirement. Available methods [26, 111, 113–116, 118–121, 207] are typically either robust or can treat multiple systems, but do not fulfill both criteria at the same time.

In this work, we propose the crystal-structure identification framework *ARISE* (ARtificial-Intelligence-based Structure Evaluation) [122]. ARISE is a specific type of ANN. The underlying theory of deep learning is discussed in section 2, in particular architecture and training of fully connected neural networks (section 2.1 and 2.2). A key component of this work is something rarely addressed in machine learning applied to materials science: quantification of model prediction uncertainty. To this end, we employ Bayesian deep neural networks, a subclass of ANNs which is discussed in section 2.3. These models do not only provide a prediction but also quantify the uncertainty, i.e., how much one can trust each prediction of the model. Specifically, we employ *Monte Carlo (MC) dropout* [123, 124], a practical way to obtain uncertainty estimates that are principled in the sense that they approximate those of a well-known probabilistic model (the *Gaussian process*). *Mutual information* allows to quantify the uncertainty.

Going beyond supervised machine learning, we employ unsupervised techniques, which are discussed in section 3. Specifically, we employ *clustering* [177, 178] (section 3.1) and *dimensionality reduction* (or rather, *manifold learning* [185], cf. section 3.2) to analyze the internal neural-network representations that are learned during training.

Given the theoretical foundations of Bayesian deep learning, chapter 4 introduces ARISE. In particular, section 4.1 summarizes which information one can extract from the classification model. In short, this comprises classification probabilities, hidden-layer representations, and uncertainty quantification. For each of these quantities, we discuss applications that demonstrate how the respective quantity can yield valuable information. The prediction workflow is explained in section 4.3: Given atomic positions labeled by chemical-species symbols, preprocessing in form of isotropic scaling is applied (section 4.3.1). Then, the materials representation is computed. In this work, we employ the *smooth-overlap-of-atomic-positions (SOAP)* descriptor [126, 127], which provides a reliable encoding of physical symmetries such as rotational invariance (cf. section 4.3.2). Specifically, we adapt SOAP such that each atomic structure is represented by a vector of fixed length, independent from the number of atoms or chemical composition. SOAP serves as input to the classification model (section 4.3.3.1), which, as discussed above, is a Bayesian neural network. Training and optimization of the model is discussed in detail in sections 4.3.3.2 and 4.3.3.3. The selection of the structural classes is the first step, which here comprises 108 different structures (a number that can be extended on demand). To demonstrate the flexibility of our approach, we select a diverse range of chemical compositions (up to four and one instance of six chemical species) and symmetries (cubic, hexagonal, tetragonal, trigonal, orthorhombic). Close-packings beyond the standard fcc (ABC) and hcp (AB) types are included, specifically double hexagonal close-packed (ABAC), and $\alpha-$Sm (ABCBCACAB). Notably, our selection traverses several material dimensions, covering bulk, 2D, and 1D materials. ARISE is not limited to predicting the space group and can classify systems where no meaningful space group number can be identified (carbon nanotubes, space group 1). To train the deep-learning model, we create a training set of $\sim$ 40k data points. To this end, we establish a protocol in which hyperparameters of SOAP are varied and periodic as well as non-periodic structures (supercells) are included. In particular, we train on different (non-periodic) supercells, since we want to investigate the local structure in large samples, which may contain structural defects such as grain boundaries and the assumption of periodic boundary conditions would lead to spurious patterns. By randomly perturbing the pristine structures, we create an extensive test set of more than $\sim$ 140k structures. Specifically, we randomly remove atoms or distort the atomic positions, while also applying these two types of defects concurrently.

Excellent performance of the trained model is demonstrated on both pristine and highly defective structures, showing 100 % classification accuracy for up to 1 % atomic displacements (defined with respect to the nearest-neighbor distance) and 10 % missing atoms. Beyond these noise levels, the accuracy drops slightly to $\sim 99\,\%$ for up to $30\,\%$ missing atoms and to $\sim 95\,\%$ for displacements of $10\,\%$. In particular, ARISE clearly outperforms the state-of-the-art methods (cf. section 5.1, in particular Tables 5.1 - 5.3). Note that no user-defined tolerance parameters are required to arrive at a satisfactory classification, which is thus fully automatic. Also note that a given atomic structure is compared to more than 100 different structural classes and the most similar prototype is reliably identified. Even for a trained materials scientist, this assignment (and the quantification of the similarity) would be a very complicated if not impossible task, in particular in case of complex periodic and possibly defective three-dimensional structures. Moreover, ARISE yields not only a classification (as it is the case for standard neural networks), but also uncertainty estimates (via the mutual information) due to the probabilistic nature of the employed Bayesian neural network.

Besides the benchmarking, section 5 discusses applications of ARISE, demonstrating how classification probabilities and uncertainty (as quantified by mutual information) can be employed for the global characterization of crystal structures. In section 5.2, we consider the first application to experimental data: highly distorted high-angle annular dark field (HAADF) images of graphene from which atomic positions are reconstructed using the deep-learning framework *AtomNet* [81]. We demonstrate that the dominating symmetry is correctly assigned (the structural class "graphene" that is included in the training set). Moreover, we find that the mutual information correlates with crystal order, e.g., its value increases if defects such as grain boundaries are contained. Section 5.3 considers the *Bain path*, a well-known structural transition path between fcc and bcc symmetries via intermediate tetragonal phases. ARISE assigns the expected prototypes along this transformation path, indicating transitions between structural classes in a smooth fashion (i.e., the classification probabilities smoothly decay at transitions, while the mutual information increases). In section 5.4, we study model performance for structures that are not included in the training set, i.e., when ARISE is forced to fail. Even though we include more than 100 structural classes, given the practically infinite number of materials, this will be a common scenario when applying ARISE. We analyze the predictions of ARISE for several out-of-sample structures and find that the assigned prototypes are physically meaningful. To this end, we inspect the input structures and the assigned structural classes, searching for matching structural characteristics (for instance, space group symmetry or bond angles). In particular, ARISE provides a ranking of most similar prototypes via the classification probability. We find that even the assignments with low classification probability are physically reasonable.

In chapter 6, we introduce the *strided pattern matching (SPM)* framework, which extends ARISE for the local characterization of large, possibly polycrystalline atomic systems. As discussed in section 6.1, we employ a local box that is scanned across the whole crystal volume. For each step, ARISE is applied, yielding a collection of classification probabilities and mutual information values. These can be rearranged into 2D (for slab-like structures) and 3D (for bulk systems) maps that reveal structural characteristics, such as grain boundaries. We first test SPM on a mono-species crystal structure with four crystalline grains in section 6.2. Each grain has a different symmetry (fcc, bcc, hcp, diamond), covering more than $80\,\%$ of the elemental solids. First, we perform a supervised analysis in section 6.2.1, i.e., we investigate the classification probabilities and the mutual information. We analyze both pristine and highly defective versions of the polycrystal and find that in both scenarios, ARISE (via SPM) assigns the correct symmetries at the expected regions. At the grain boundaries, the mutual information increases while being low within the crystalline grains. Going beyond the supervised analysis, we conduct an unsupervised study of the same structure in section 6.2.2. We employ state-of-the-art clustering and dimensionality-reduction algorithms for the analysis of the internal representations that are learned by the neural-network model. Specifically, we investigate the internal representations of the local boxes of the mono-species polycrystal. We find that the cluster assignments (obtained via Hierarchical Density-based Spatial Clustering Applications with Noise, short *HDBSCAN*) are physically meaningful as they correspond to the four crystalline grains (while HDSCAN also detects outliers, which here correspond to grain-boundary and outer-border regions). Via manifold learning (Uniform Manifold Approximation and Projection, short *UMAP*), we project the high-dimensional representations into 2D, providing an easily interpretable structural-similarity map. Even in this highly compressed representation, points that correspond to the same crystalline grain and thus are similar to each other, are grouped. In this synthetic setting, we can also compare the predictions of ARISE with the cluster assignments, which turn out to be closely matching, thus verifying the established unsupervised-learning protocol. Following the notions on *interpretability* and *explainability* established in [42–44], this unsupervised study represents a *post hoc explanation* of the trained model. In section 6.3, we consider a more realistic synthetic structure, which serves as a model system of Ni-based superalloys. The task is to detect an ordered precipitate ($L1_2$ symmetry) in a random fcc matrix. ARISE (via SPM) accomplishes this task, even in presence of high levels of noise (20 % missing atoms, 5 % atomic displacements). Notably, the results demonstrate that ARISE can be used to analyze chemical ordering, since matrix and precipitate share the same lattice sites and only differ in chemical distribution. This characteristic is useful, for instance, in atomic-electron-tomography (AET) experiments on FePt nanoparticles. In section 6.4, an experimentally validated grain-boundary system that has been generated via evolutionary structure search is investigated. Compared to the previous synthetic structures, this system is a result from atomistic simulations (specifically an extension of the crystal-structure prediction method *USPEX* [220]). We show that ARISE can detect the correct symmetry in the grains and also provides physically meaningful predictions at the grain boundary. Section 6.5 discusses the investigation of HAADF images of graphene, i.e., similar to section 5.2, while here, the structures are investigated on the local scale. In particular, we demonstrate that grain boundaries can be detected. To demonstrate the flexibility of our approach (i.e., the combination of ARISE, SPM and reconstruction of atomic positions or rather columns from 2D electron-microscopy data), we study a high-resolution transmission electron microscopy (HRTEM) image of a quasicrystal. ARISE and SPM allow to detect the dominating pattern, thus confirming the established classification protocol and even revealing an interesting – yet under-explored – future research path: the automatic classification of quasicrystals. Going from 2D to 3D experimental data, we consider an AET study of FePt nanoparticles in section 6.6. Similar to the synthetic mono-species polycrystal, we test both supervised and unsupervised-learning protocols. In section 6.6.1, the SPM framework assigns structural classes that are in line with physical intuition (i.e., certain phases are expected to appear for these kinds of systems). Then, we apply unsupervised learning to the same structure in section 6.6.2. Compared to section 6.2.2, in this setting, no ground truth is known and the data is coming from experiment, i.e., the analysis is truly exploratory. We first extract all boxes that are classified as the tetragonal phase $L1_0$ due to its technological relevance. Then we apply the clustering algorithm HDBSCAN. We find that the cluster assignments can be explained by a specific, easily interpretable geometrical property that is differently distributed among the clusters. This study provides an example of how one can perform a machine-learning driven, exploratory analysis of structural data. Finally, adding a further dimension, we investigate 4D AET data, i.e., a time-resolved measurement of FePt nanoparticles in section 6.7. Here, we find that the mutual information correlates with the crystal order over time. Together with the previous studies (for instance, on grain-boundary detection), this supports that mutual information qualifies as an AI-based order parameter.

Chapter 8

Conclusions

In this work, Bayesian deep learning is employed to achieve a flexible, robust and threshold-independent crystal classification model, which we term ARISE. This approach correctly classifies a comprehensive and diverse set of crystal structures from computations and experiments, including polycrystalline systems (via strided pattern matching). ARISE is trained on ideal synthetic systems only and correctly identifies crystal structures in scanning transmission electron microscopy (STEM), HRTEM, and AET experiments, hence demonstrating strong generalization capabilities. Notably, we traverse multiple dimensions from 2D (graphene) to 3D (FePt nanoparticles) and even 4D (time-resolved, 3D analysis of FePt nanoparticles). The Bayesian-deep-learning model provides classification probabilities, which – at variance with standard neural networks – allow for the quantification of predictive uncertainty via mutual information. The mutual information is found to directly correlate with the degree of crystalline order, as shown by the analysis of time-resolved data from AET experiments. The internal neural-network representations are analyzed via state-of-the-art unsupervised learning. The clusters identified in this high-dimensional internal space allow to uncover physically meaningful structural regions. These can be grain boundaries, but also unexpected substructures sharing geometrical properties, as shown for metallic nanoparticles from AET experiments. This illustrates how supervised and unsupervised machine learning can be combined to discover hidden patterns in materials science data.

Given limited amounts of data, ARISE and SPM provide an interesting pathway to generate and analyze large quantities of hidden and valuable information. For instance, when considering AET data comprising around 23k atoms (cf. Fig. 6.8, 6.9), a standard procedure is to center the local atomic environments at the atoms and classify each of them into a list of N reference structures. In ARISE, N equals 108, and thus already at this level, even if adopting the standard definition of local atomic environments, the amount of information is often increased by two orders of magnitude in comparison to state-of-the-art methods (cf. section 5.1). Moreover, we introduce a sliding box in SPM to study the transition between local atomic environments in smooth fashion. This procedure increases the number of local atomic environments even further: in case of the mentioned AET data, the information from 23k environments is almost doubled to $\sim$ 44k for a stride of 2 Å, while for smaller stride the information is growing even further. Besides the 108 classification probabilities, the uncertainty can be quantified which provides a novel way to assess the degree of structural order. In addition, the hidden layer representations can be studied, providing additional, high-dimensional vectors $\mathbf{h}_1 \in \mathbb{R}^{256}, \mathbf{h}_2 \in \mathbb{R}^{512}$, and $\mathbf{h}_3 \in \mathbb{R}^{256}$, where each of these components can contain valuable information. This may be analyzed with unsupervised learning (clustering, dimensionality reduction), leading to additional information in form of cluster assignments or distances within structural-similarity maps. To explain, for instance, the cluster assignments, geometrical properties can be calculated (nearest-neighbor distances, angle distribution, coordination number). These explanations do not only provide insight into the trained model but can be used to gain physical insights. For instance, we demonstrate an example of an exploratory analysis of experimental data in section 6.6.2.

While we do provide *post hoc* explanations, model *transparency* [42] is hindered due to the black box nature of the Bayesian-deep-learning model. Moreover, for the presented model, there is no strategy available to relate input features and final predictions in order to discover physically meaningful connections. This is mainly due to the use of the SOAP descriptor, whose components correspond to basis set expansion coefficients, from which intuitive explanations are hard to infer.

To obtain a robust model, a sufficiently large training set has to be created and the corresponding labels for the crystallographic prototypes need to be known. This information may not be in general available; in this work, databases such as AFLOW and NOMAD represent an essential resource.

Given its data-driven nature, ARISE is not limited to predicting the space group. Thus systems where the space group does not characterize the crystal structure can be tackled (as demonstrated for carbon nanotubes). More complex systems such as quasi-crystals [226], periodic knots, or weavings [240] could also be considered. Indeed, ARISE can be applied to any data (computational or experimental) providing Cartesian coordinates labeled by chemical species. Practically, one simply needs to add the new structures of interest to the training set, and re-train or fine-tune (i.e., via transfer learning) the neural network with the desired labels. In analogy with the illustrative example of the Bain path (cf. section 5.3), ARISE can be readily applied to study any structural transitions (provided that the structures of interests are included in the 108 classes considered in this work). Moreover, the mutual information allows to quantify the defectiveness of a structure; this could be exploited to automatically evaluate the quality of STEM images. For example, one may automatically screen for STEM images that are likely to contain structural defects. Applications in active learning [241] for materials science are also envisioned, where uncertainty is crucial, for example, when deciding on the inclusion of additional - typically computationally costly - points in the dataset. Furthermore, it would be interesting to change the definition of classes for ARISE to compare it with specialized classification schemes identifying the Bravais lattice [242] or performing topological classifications [243]. Moreover, future research could use neural networks to classify defects [244] or to predict continuous properties (for instance, energy or band gap) together with crystal-structure labels.

The experimental applications in this work concentrated on electron-microscopy techniques, in particular HAADF and HRTEM images, as well as electron tomography (specifically, AET) data. Atom probe tomography (APT) is another technique that provides atomic resolution, while being much more limited in its spatial resolution. Although in some regions sub-Å resolution can be reached [245] and lattice planes resolved [246], typically the amount of in-plane experimental noise results in 3D atomic arrangements resembling more amorphous structures than ordered crystalline structures. Although ARISE is shown to be robust beyond physically reasonable noise levels and substantially improves the state of the art in crystal-structure recognition, the amount of distortions typically present in atom-probe tomography experiments is currently out of reach. Still, ARISE is arguably the most promising method for analyzing 3D crystallographic information in atom-probe data. Additional improvements on robustness are possible in the future: ARISE flexibility allows to train on (highly) defective structures. For this purpose, training on randomly perturbed structures is only a first test and more complex noise models have to be constructed. Simulations of the APT experiment [247–249] or atom-probe informed simulations [250, 251] might be a useful data source.

The STEM and HRTEM analyses presented in sections 6.5 and 5.2 provide a blueprint for the automatic study of image-like microscopy data: given an atomic-resolution image, one reconstructs the atomic positions and uses ARISE for global and local characterization (specifically, to detect the underlying, most similar symmetry or identify defects such as grain boundaries). An important step in this procedure is the reconstruction of atomic positions. Here, we employ the deep-learning framework AtomNet, which is specialized to highly distorted HAADF images of graphene but can be extended to other systems. For instance, extensions to cubic systems and the detection of Si defects in graphene (alongside chemical-species classification) have been demonstrated [84]. One could also employ ARISE for improving the reconstruction, using, for instance, the mutual information as a feedback variable (quantifying structural order of reconstructed atomic positions) to guide a learning algorithm.

A further interesting application of ARISE is screening large pools of atomic structures. These structures may be extracted from computational repositories such as AFLOW and NOMAD, but may also come from atomistic simulations. For instance, the crystal-structure prediction method USPEX has been extended to study systems containing grain boundaries [252]. In [98], a correlation between physical properties (grain-boundary stress and volume) and structural features has been found (so-called "Domino" and "Pearl" patterns at the grain boundary). However, no 3D structural information has been taken into account in this study – valuable information that ARISE can quantify. Note that in section 6.4, we analyzed one of the structures calculated in [98], demonstrating how ARISE can be applied to these kinds of systems. One may even employ ARISE to improve the crystal-structure prediction algorithm itself, by screening the structures that are created in each iteration of the evolutionary algorithm that underlies USPEX. For instance, one may separate the pool of generated structures into ordered and disordered systems using mutual information or filter out specific structural classes using the assignments of ARISE.

Appendix A

Benchmarking

This section reports benchmarking studies comparing ARISE to state-of-the-art methods, complementing the results reported in section 5.1.

	Pristine	Random displacements (δ)					Missing atoms (η)			
		0.1%	0.6%	1%	2%	4%	1%	5%	10%	20%
Spglib (loose)	100.00	100.00	100.00	95.26	0.20	0.00	11.23	0.00	0.00	0.00
Spglib* (loose)	67.71	67.71	67.71	65.83	14.51	0.00	15.73	0.03	0.00	0.00
Spglib (tight)	100.00	0.00	0.00	0.00	0.00	0.00	11.23	0.00	0.00	0.00
Spglib* (tight)	83.33	0.00	0.00	0.00	0.00	0.00	17.53	0.00	0.00	0.00
PTM	100.00	100.00	100.00	100.00	100.00	100.00	88.67	51.76	25.93	6.24
PTM*	8.78	11.37	11.37	11.37	11.37	11.37	10.08	5.90	2.96	0.71
CNA	66.14	62.81	62.81	54.55	32.34	31.41	55.86	32.50	15.75	3.07
CNA*	1.44	1.62	1.62	1.40	0.83	0.81	1.44	0.84	0.41	0.08
a-CNA	100.0	100.0	100.0	100.0	100.0	100.0	89.25	52.81	25.92	5.37
a-CNA*	2.49	3.08	3.08	3.08	3.08	3.08	2.75	1.64	0.81	0.17
BAA	100.0	100.0	100.0	100.0	100.0	97.85	99.71	88.78	65.21	25.38
BAA*	2.49	3.08	3.08	3.08	3.08	3.03	3.08	2.74	2.02	0.81
ARISE (108 / 108)	100.00	100.00	100.00	100.00	99.86	99.29	100.00	100.00	100.00	99.85

Table A.1: Accuracy in identifying the parent class of defective crystal structures. Two lines are shown for each of the methods used for benchmarking (spglib, PTM, CNA, a-CNA, BAA): In rows without stars, the accuracy is calculated only for structures for which the respective method was designed for; for instance, spglib can be applied to every structure of Fig. 6.1e except the 12 nanotubes (note that we only include prototypes from AFLOW for spglib, cf. section 5.1). This is also true for the other methods, while additional structures have to be removed, for instance, for CNA, a-CNA, and BAA as they cannot classify simple cubic and diamond structures. In starred rows, all 108 classes summarized in Fig. 6.1e are included, leading to the strong decrease in performance. In contrast, the neural-network approach proposed here can be applied to all classes, and thus the whole dataset was used.

	Random displacements (δ)		Missing atoms (η)	
	7%	10%	25%	30%
Spglib (loose)	0.00	0.00	0.00	0.00
Spglib* (loose)	0.00	0.00	0.00	0.00
Spglib (tight)	0.00	0.00	0.00	0.00
Spglib* (tight)	0.00	0.00	0.00	0.00
PTM	100.00	94.34	3.33	1.72
PTM*	11.37	10.71	0.38	0.19
CNA	31.41	24.20	1.38	0.55
CNA*	0.81	0.62	0.04	0.01
a-CNA	99.99	94.55	2.60	1.03
a-CNA*	3.08	2.90	0.08	0.03
BAA	87.79	69.68	14.25	7.35
BAA*	2.77	2.22	0.49	0.30
ARISE (this work)	97.82	94.56	99.86	99.76

Table A.2: Accuracy in identifying the parent class of defective crystal structures for high displacements (percentage δ) and missing atoms (percentage η).

	Missing atoms and displacements (η, δ)						
	(1%, 0.1%)	(5%, 0.6%)	(10%, 1%)	(15%,2%)	(20%,4%)	(25%,7%)	(30%,10%)
Spglib (loose)	11.32	0.00	0.00	0.00	0.00	0.00	0.00
Spglib* (loose)	15.76	0.00	0.00	0.00	0.00	0.00	0.00
Spglib (tight)	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Spglib* (tight)	0.00	0.00	0.00	0.00	0.00	0.00	0.00
PTM	88.68	51.78	25.60	12.75	6.41	3.19	1.46
PTM*	10.08	5.90	2.92	1.45	0.73	0.36	0.16
CNA	55.77	31.95	13.83	4.41	2.03	0.79	0.19
CNA*	1.44	0.82	0.36	0.11	0.05	0.02	0.00
a-CNA	89.21	52.36	26.01	12.13	6.07	2.40	0.97
a-CNA*	2.75	1.62	0.81	0.38	0.19	0.08	0.03
BAA	99.72	88.98	65.17	42.62	25.95	15.58	6.63
BAA*	3.07	2.75	2.02	1.34	0.82	0.50	0.22
ARISE (this work)	100.00	100.00	100.00	99.88	99.29	97.31	92.50

Table A.3: Accuracy in identifying the parent class of defective crystal structures, with both missing atoms (percentage η) and displacements (percentage δ) introduced at the same time. The results show that ARISE is also robust for highly defective structures where displacements and missing atoms are present at the same time. This is the typical situation encountered in APT – making ARISE arguably the best available candidate for crystal-structure classification in APT experimental data.

Appendix B

SOAP descriptor

This section contains a study (Fig. B.1) that provides additional details on the change of the SOAP descriptor with supercell size.

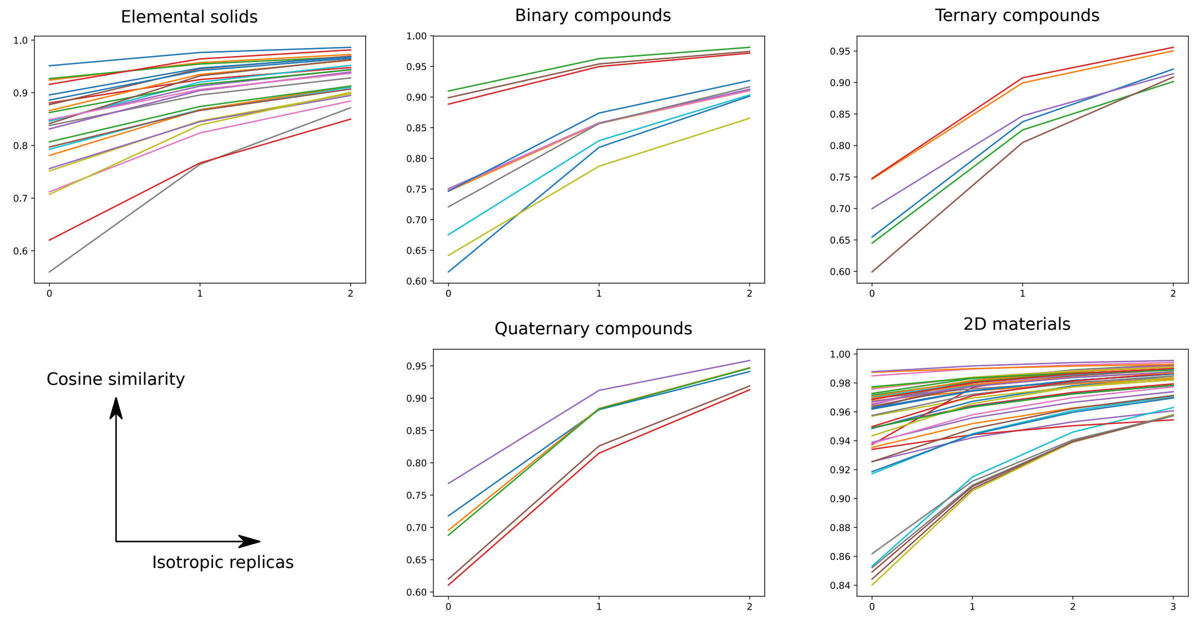

Figure B.1: Cosine similarity plots for elemental, binary, ternary, and quaternary compounds as well as 2D materials (for SOAP settings $R_C = 4.0 \cdot d_{NN}$, $\sigma = 0.1 \cdot d_{NN}$, and extrinsic scaling factor = 1.0 corresponding to the center of the parameter range used in the training set). Each line corresponds to a particular prototype. The x-axis corresponds to three different (non-periodic) supercells, where supercell “0” stands for the smallest isotropic supercell (for instance, $4 \times 4 \times 4$ repetitions) for which at least 100 atoms are obtained. Supercells “1” and “2” correspond to the next two bigger isotropic replicas (e.g., $5 \times 5 \times 5$ and $6 \times 6 \times 6$). The y-axis corresponds to the cosine similarity of the respective supercell to the periodic structure, i.e., the case of infinite replicas. One can see a continuous increase of similarity with larger supercell size, where for the largest supercell, the similarity is greater than 0.9 for all prototypes. Thus, it is to be expected that systems sizes larger than those included in the training set can be correctly classified by ARISE. For smaller systems, however, generalization ability will depend on the prototype. Practically, one can include smaller supercells in the training set, which is not a major problem due to fast convergence time.

Appendix C

Bain path

This section contains a figure that supports the Bain-path study discussed in section 5.3.

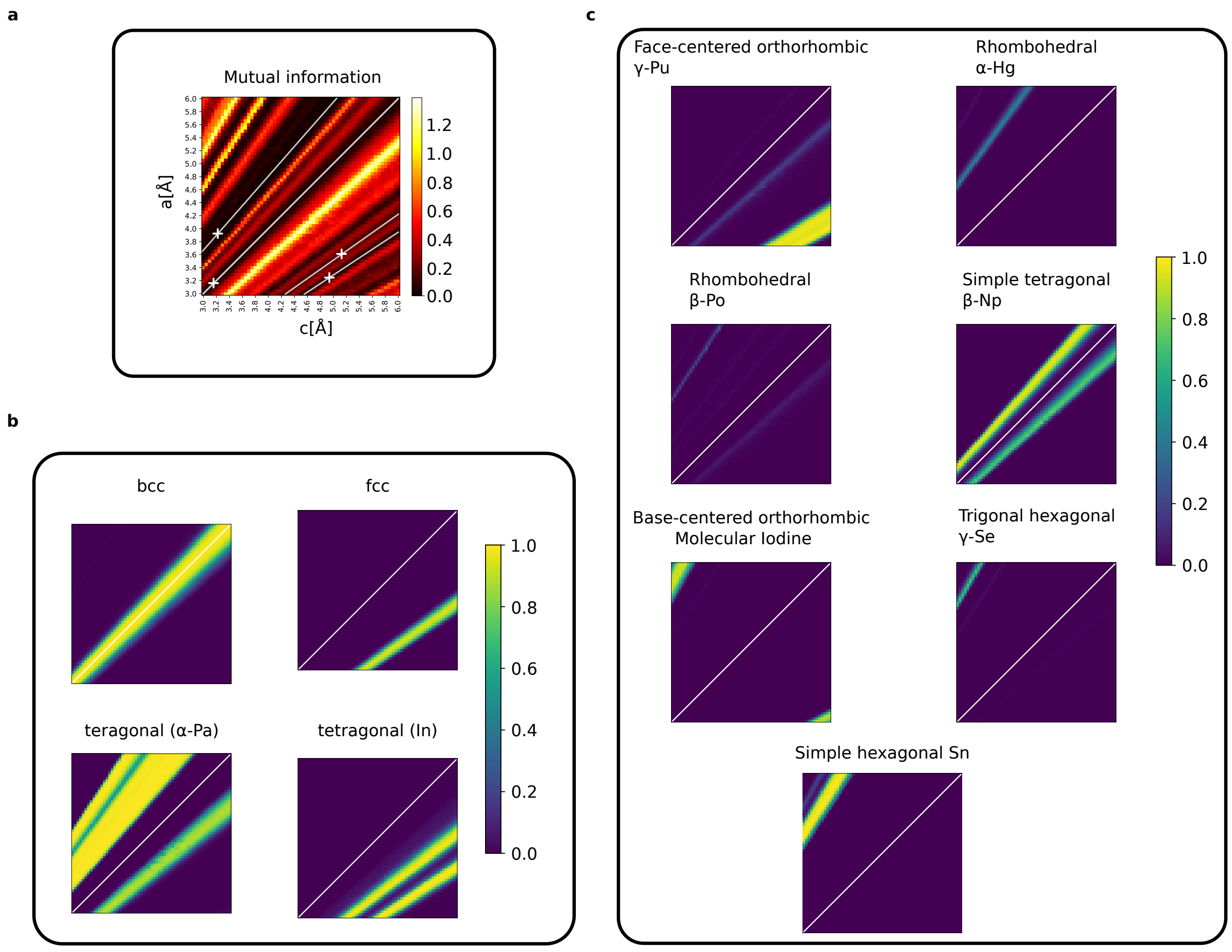

Figure C.1: Bain path - all prototypes with increased classification probability: **a** Mutual information plot showing the spots of high and low uncertainty for different geometries. **b** Classification probability maps corresponding to bcc, fcc and two tetragonal phases. **c** Representative selection of other prototypes showing non-zero classification probabilities.

Appendix D

Local analysis via SPM and ARISE

This section contains several figures that support the analysis of the mono-species system (via ARISE and SPM) discussed in section 6.2. Specifically, for the supervised analysis (cf. section 6.2.1), Fig. D.2, and Fig. D.3 provide additional details (see the respective figure captions), while Fig. D.1 and Fig. D.4 support the unsupervised analysis discussed in section 6.2.2.

Finally, the analysis of AET data is supported by two figures (Fig. D.5 and Fig. D.6).

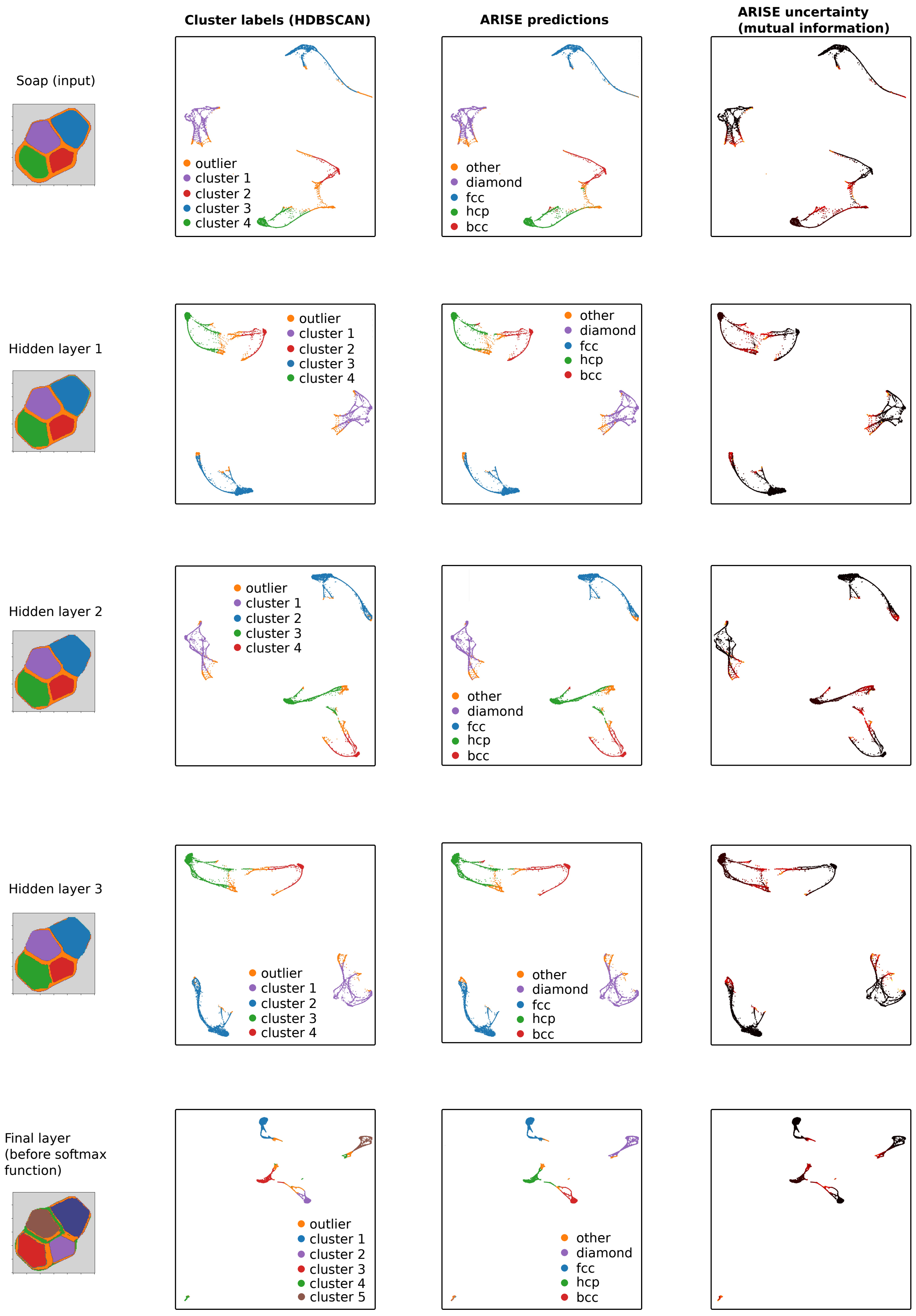

Figure D.1: Unsupervised analysis analogous to Figure 6.3, for all layers (before the ReLU or rather the softmax function).

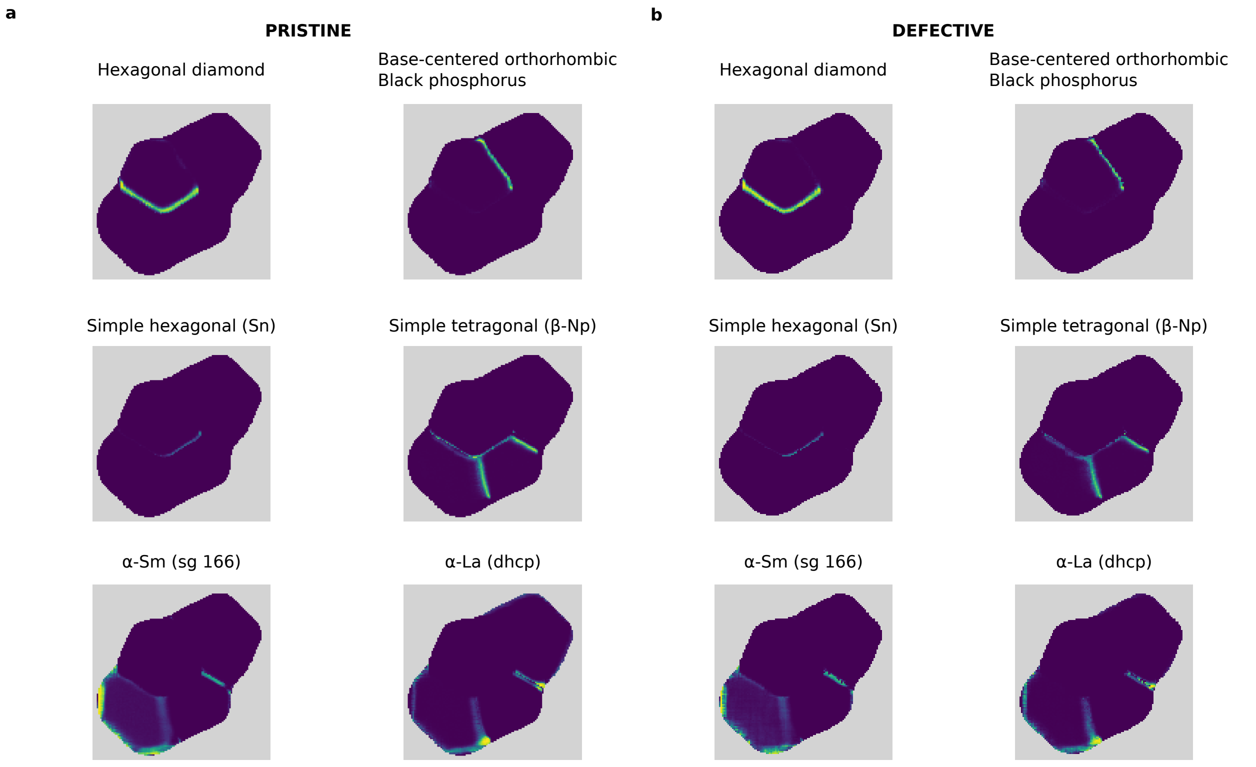

Figure D.2: Probability maps of the most important prototypes for both pristine (**a**) and defective (**b**) mono-species polycrystal.

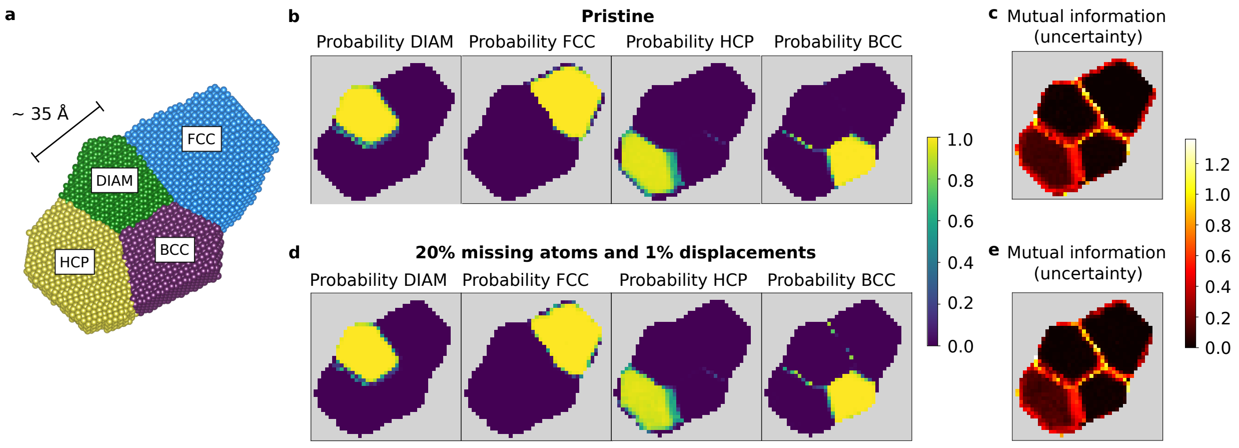

Figure D.3: Mono-species elemental polycrystal investigation via strided pattern matching using lower resolution, specifically a stride of 3.0 Å in both x and y direction opposed to 1.0 Å as in Fig. 6.2. Choosing the stride is a trade-off between computation time and resolution. For instance, at the grain boundary between diamond and hcp, the transition from diamond to hexagonal diamond to hcp (cf. D.2) are recognized in Fig. 6.2, while being obscured in the presented low-resolution pictures.

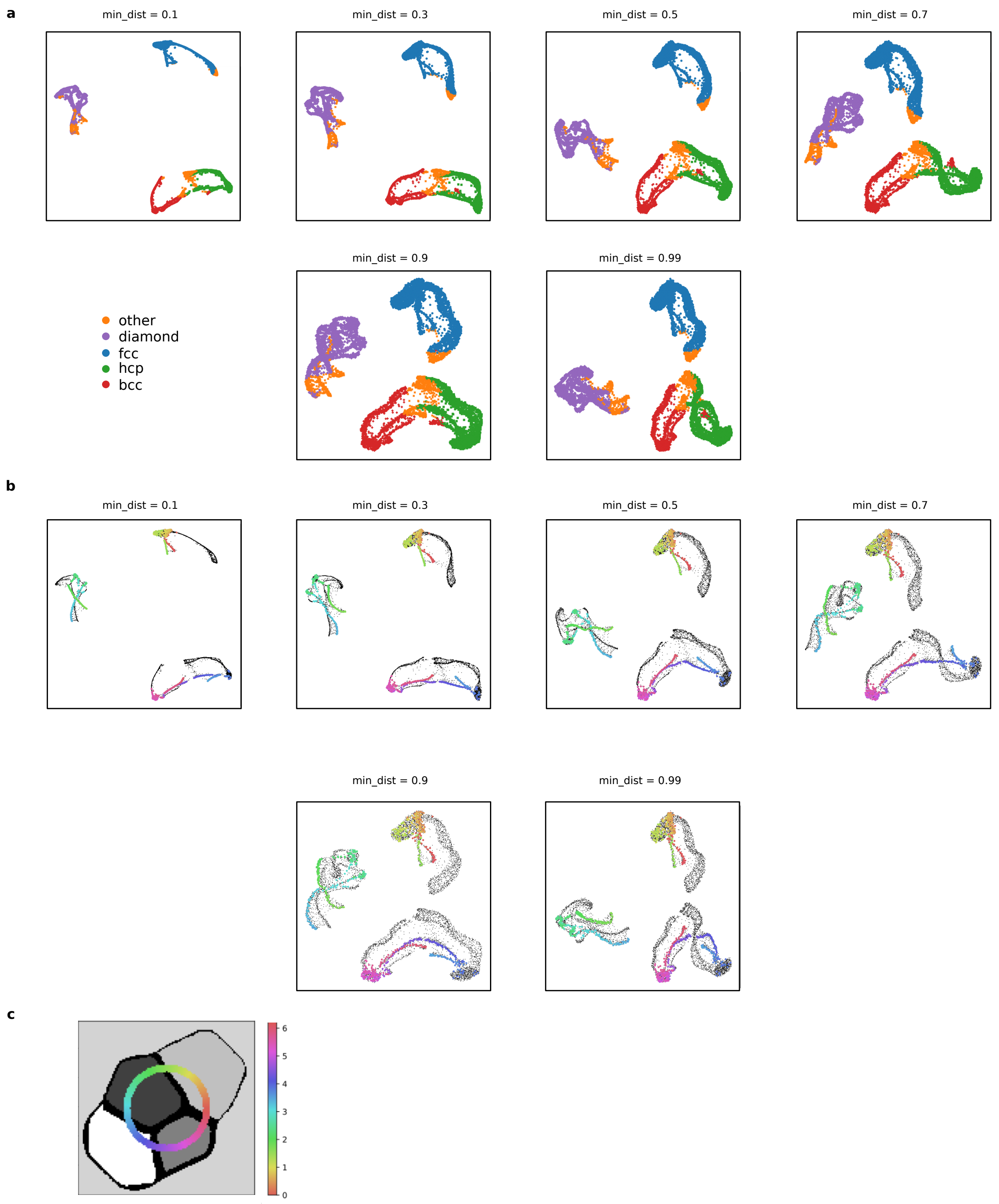

Figure D.4: Connection between UMAP embedding and real space. This figure confirms the observation that ARISE's representations of different spatial regions (crystalline regions but in particular grain boundaries, here: transitions between fcc, bcc, hcp, and diamond) are mapped to different regions in the UMAP projection. **a** Influence of the minimum distance parameter ("min_dist", cf. the final paragraph in section 3.2.1) in the UMAP projection (number of neighbors fixed to 500). In line with intuition, for larger min_dist, points appear more spread. In particular, connected subregions appear in the clusters, whose connection to real space is investigated in **b**: The connected strings of points actually correspond to transitions within and between crystalline regions. This is demonstrated by traversing a circle around the center of the real space structure (**c**) and coloring the embedded points according to the angle (in radian).

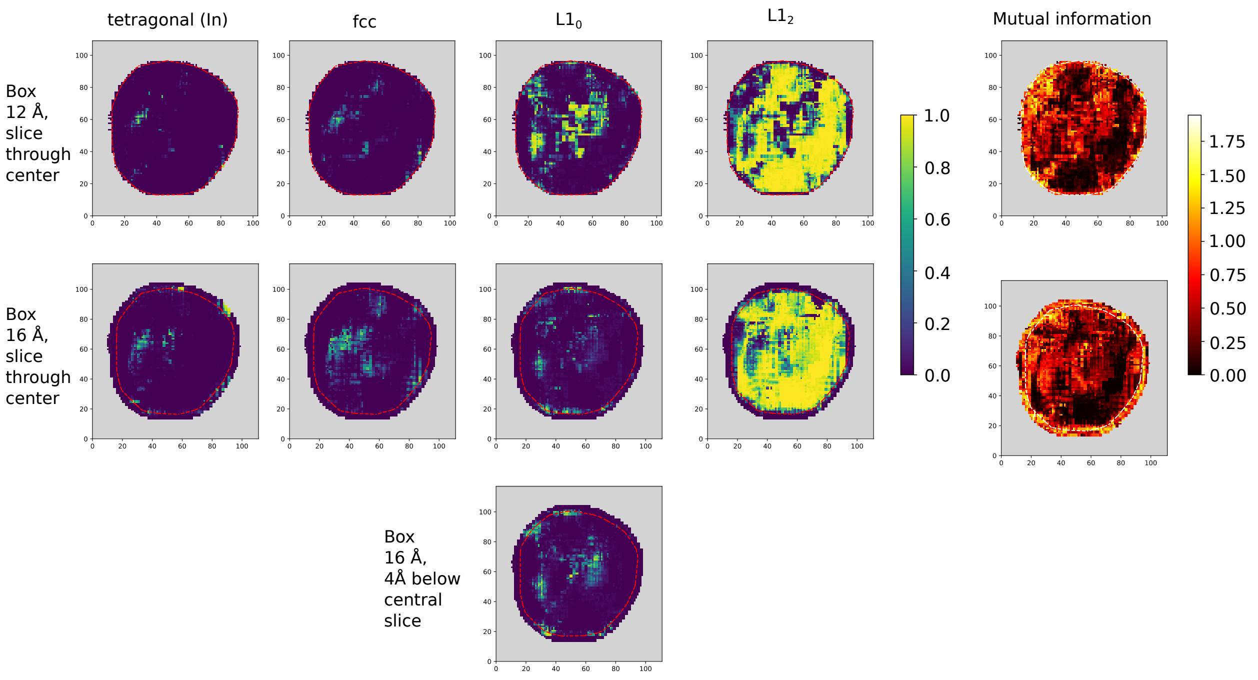

Figure D.5: Comparison of crystal maps (slice through center, most important prototypes and mutual information) for AET nanoparticle data [237] for two different box sizes. Dashed lines indicate the crystal boundaries in all 2D maps. ARISE allows to investigate the appearance of the tetragonally distorted fcc prototype (In). For larger box sizes, the fcc assignment increases in the center and also the L1$_2$ classification probability rises. While the central slice of the L1$_0$ prototype for a box size of 16 Å shows only weak signal, a slice slightly below reveals higher probability (see bottom, isolated slice), i.e., ARISE does not overlook this technologically relevant phase.

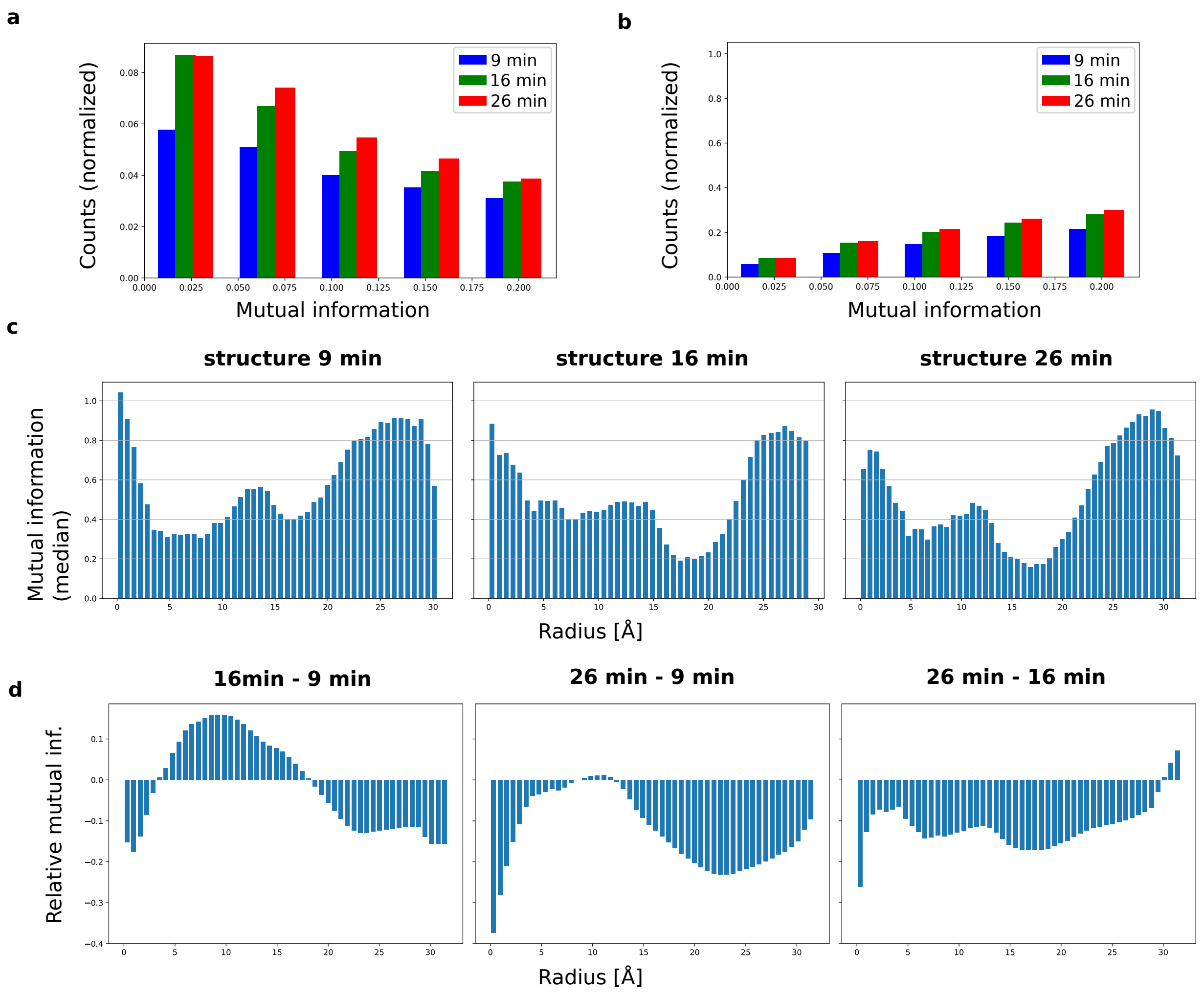

Figure D.6: Quantitative study of mutual information distribution for different annealing times. **a** Histogram of mutual information values for each annealing time (where the corresponding histograms are normalized via dividing each bin by the total number of boxes). Only mutual-information values smaller than 0.2 are shown, which correspond to the ”dark”, i.e., low mutual information spots in Fig. 6.10**a**. **b** Cumulative distribution calculated for the histogram shown in **a**. From **a**, **b** it is apparent that the number of low-uncertainty boxes increases for larger annealing times. **c-d** Investigation of the radial distribution of the mutual information. **c** Histograms of uncertainty (mutual information) obtained via spatially binning the SPM maps of 6.10**a** into spherical shells, where the median is computed for each bin. Given a mutual information value, the associated radius is calculated as the distance of the center of the corresponding box (as obtained via SPM) to the center of the most central box. **d** Each panel shows the difference between the cumulative distributions of two annealing times, where the cumulative distributions are calculated from the histograms shown in **b**. In addition the histograms are normalized the following way: Given the times t_1, t_2 with $t_1 < t_2$, the cumulative sum of $t_2 - t_1$ is calculated and then divided by the cumulative sum of time t_1 such that the fractional change from t_1 to t_2 is obtained. One can conclude that in **c** a clear decrease of mutual information can be spotted in specific regions, e.g., for the radial region 15-20 Å. The cumulative sums that are used in **d** allow to quantify the order more globally in the sense that each bin (of the cumulative sum corresponding to a specific annealing time) is proportional to the spherically averaged integral from radius zero up to the radius corresponding to the bin. Since the particle sizes are changing over time due to diffusion, the particles have different size. Thus, we single out a radius at which to compare the global order: for instance, comparing the bins corresponding to a radius of r=25 Å, we see that for all three panels, the values are negative and thus the structure that has been annealed longer shows larger global order.

Bibliography

[1] S. Russell and P. Norvig. *Artificial intelligence: a modern approach* (Prentice Hall, 2002).

[2] M. Haenlein and A. Kaplan. *A brief history of artificial intelligence: On the past, present, and future of artificial intelligence.* California Management Review **61**, 5 (2019).

[3] G. Hager and G. Wellein. *Introduction to high performance computing for scientists and engineers* (CRC Press, 2010).

[4] J. D. Owens, M. Houston, D. Luebke, S. Green, J. E. Stone and J. C. Phillips. *GPU computing.* Proceedings of the IEEE **96**, 879 (2008).

[5] N. P. Jouppi, C. Young, N. Patil, D. Patterson, G. Agrawal, R. Bajwa, S. Bates, S. Bhatia, N. Boden, A. Borchers et al. *In-datacenter performance analysis of a tensor processing unit.* In *Proceedings of the 44th Annual International Symposium on Computer Architecture*, 1–12 (2017).

[6] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li and L. Fei-Fei. *Imagenet: A large-scale hierarchical image database.* In *2009 IEEE conference on computer vision and pattern recognition*, 248–255 (2009).

[7] L. Deng. *The MNIST database of handwritten digit images for machine learning research.* IEEE Signal Processing Magazine **29**, 141 (2012).

[8] D. E. Rumelhart, G. E. Hinton and R. J. Williams. *Learning representations by back-propagating errors.* Nature **323**, 533 (1986).

[9] X. Glorot, A. Bordes and Y. Bengio. *Deep sparse rectifier neural networks.* In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, 315–323 (JMLR Workshop and Conference Proceedings, 2011).

[10] D. P. Kingma and J. Ba. *Adam: A method for stochastic optimization.* arXiv preprint arXiv:1412.6980 (2014).

[11] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville and Y. Bengio. *Generative adversarial nets.* Advances in Neural Information Processing Systems **27**, 2672 (2014).

[12] Y. LeCun, Y. Bengio and G. Hinton. *Deep learning.* Nature **521**, 436 (2015).

[13] J. Schmidhuber. *Deep learning in neural networks: An overview.* Neural Networks **61**, 85 (2015).

[14] I. Goodfellow, Y. Bengio and A. Courville. *Deep Learning* (MIT Press, 2016).

[15] A. Krizhevsky, I. Sutskever and G. E. Hinton. *Imagenet classification with deep convolutional neural networks.* In *Advances in Neural Information Processing Systems*, 1097–1105 (2012).

[16] G. Hinton, L. Deng, D. Yu, G. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, B. Kingsbury et al. *Deep neural networks for acoustic modeling in speech recognition.* IEEE Signal processing magazine **29** (2012).

[17] I. Sutskever, O. Vinyals and Q. V. Le. *Sequence to sequence learning with neural networks.* In *Advances in Neural Information Processing Systems*, 3104–3112 (2014).

[18] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot et al. *Mastering the game of Go with deep neural networks and tree search.* Nature **529**, 484 (2016).

[19] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton et al. *Mastering the game of go without human knowledge.* Nature **550**, 354 (2017).

[20] G. Carleo, I. Cirac, K. Cranmer, L. Daudet, M. Schuld, N. Tishby, L. Vogt-Maranto and L. Zdeborová. *Machine learning and the physical sciences.* Reviews of Modern Physics **91**, 045002 (2019).

[21] J. Schmidt, M. R. Marques, S. Botti and M. A. Marques. *Recent advances and applications of machine learning in solid-state materials science.* npj Computational Materials **5**, 1 (2019).

[22] A. J. Hey, S. Tansley, K. M. Tolle et al. *The fourth paradigm: data-intensive scientific discovery*, volume 1 (Microsoft research Redmond, WA, 2009).

[23] C. Draxl and M. Scheffler. *NOMAD: The FAIR concept for big data-driven materials science.* MRS Bulletin **43**, 676 (2018).

[24] C. Draxl and M. Scheffler. *The NOMAD laboratory: from data sharing to artificial intelligence.* Journal of Physics: Materials **2**, 036001 (2019).

[25] C. Draxl and M. Scheffler. *Big data-driven materials science and its FAIR data infrastructure.* Handbook of Materials Modeling: Methods: Theory and Modeling 49–73 (2020).

[26] A. Ziletti, D. Kumar, M. Scheffler and L. M. Ghiringhelli. *Insightful classification of crystal structures using deep learning.* Nature Communications **9**, 1 (2018).

[27] B. Goertzel and C. Pennachin. *Artificial general intelligence*, volume 2 (Springer, 2007).

[28] S. Legg, M. Hutter et al. *A collection of definitions of intelligence.* Frontiers in Artificial Intelligence and applications **157**, 17 (2007).

[29] S. Legg and M. Hutter. *Universal intelligence: A definition of machine intelligence.* Minds and Machines **17**, 391 (2007).

[30] F. Chollet. *On the measure of intelligence.* arXiv preprint arXiv:1911.01547 (2019).

[31] K. P. Murphy. *Machine learning: a probabilistic perspective* (MIT press, 2012).

[32] C. M. Bishop. *Pattern recognition and machine learning* (Springer, 2006).

[33] J. Friedman, T. Hastie, R. Tibshirani et al. *The elements of statistical learning*, volume 1 (Springer, 2001).

[34] G. James, D. Witten, T. Hastie and R. Tibshirani. *An introduction to statistical learning*, volume 112 (Springer, 2013).

[35] A. L. Samuel. *Some studies in machine learning using the game of checkers. II—Recent progress.* IBM Journal of Research and Development **11**, 601 (1967).

[36] J. R. Koza, F. H. Bennett, D. Andre and M. A. Keane. *Automated design of both the topology and sizing of analog electrical circuits using genetic programming.* 151–170 (Springer, 1996).

[37] T. Mitchell. *Machine Learning* (McGraw-Hill, 1997).

[38] P. Domingos. *A few useful things to know about machine learning.* Communications of the ACM **55**, 78 (2012).

[39] L. M. Ghiringhelli, J. Vybiral, S. V. Levchenko, C. Draxl and M. Scheffler. *Big data of materials science: critical role of the descriptor.* Physical Review Letters **114**, 105503 (2015).

[40] O. Isayev, C. Oses, C. Toher, E. Gossett, S. Curtarolo and A. Tropsha. *Universal fragment descriptors for predicting properties of inorganic crystals.* Nature Communications **8**, 15679 (2017).

[41] T. Xie and J. C. Grossman. *Crystal graph convolutional neural networks for an accurate and interpretable prediction of material properties.* Physical Review Letters **120**, 145301 (2018).

[42] Z. C. Lipton. *The Mythos of Model Interpretability: In machine learning, the concept of interpretability is both important and slippery.* Queue **16**, 31 (2018).

[43] R. Roscher, B. Bohn, M. F. Duarte and J. Garcke. *Explainable machine learning for scientific insights and discoveries.* IEEE Access **8**, 42200 (2020).

[44] W. J. Murdoch, C. Singh, K. Kumbier, R. Abbasi-Asl and B. Yu. *Definitions, methods, and applications in interpretable machine learning.* Proceedings of the National Academy of Sciences **116**, 22071 (2019).

[45] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction* (MIT press, 2018).

[46] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel et al. *Mastering chess and shogi by self-play with a general reinforcement learning algorithm.* arXiv preprint arXiv:1712.01815 (2017).

[47] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel et al. *A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play.* Science **362**, 1140 (2018).

[48] J. Schrittwieser, I. Antonoglou, T. Hubert, K. Simonyan, L. Sifre, S. Schmitt, A. Guez, E. Lockhart, D. Hassabis, T. Graepel et al. *Mastering atari, go, chess and shogi by planning with a learned model.* Nature **588**, 604 (2020).

[49] M. Popova, O. Isayev and A. Tropsha. *Deep reinforcement learning for de novo drug design.* Science Advances **4**, eaap7885 (2018).

[50] T. M. Dieb, S. Ju, K. Yoshizoe, Z. Hou, J. Shiomi and K. Tsuda. *MDTS: automatic complex materials design using Monte Carlo tree search.* Science and technology of advanced materials **18**, 498 (2017).

[51] M. Olivecrona, T. Blaschke, O. Engkvist and H. Chen. *Molecular de-novo design through deep reinforcement learning.* Journal of cheminformatics **9**, 1 (2017).

[52] A. Zhang, Z. C. Lipton, M. Li and A. J. Smola. *Dive into Deep Learning* (2020). https://d2l.ai.

[53] S.-M. Udrescu and M. Tegmark. *AI Feynman: A physics-inspired method for symbolic regression.* Science Advances **6**, eaay2631 (2020).

[54] R. Ouyang, S. Curtarolo, E. Ahmetcik, M. Scheffler and L. M. Ghiringhelli. *SISSO: A compressed-sensing method for identifying the best low-dimensional descriptor in an immensity of offered candidates.* Physical Review Materials **2**, 083802 (2018).

[55] R. Ouyang, E. Ahmetcik, C. Carbogno, M. Scheffler and L. M. Ghiringhelli. *Simultaneous learning of several materials properties from incomplete databases with multi-task SISSO.* Journal of Physics: Materials **2**, 024002 (2019).

[56] J. W. Tukey. *We need both exploratory and confirmatory.* The American Statistician **34**, 23 (1980).

[57] L. Ward and C. Wolverton. *Atomistic calculations and materials informatics: A review.* Current Opinion in Solid State and Materials Science **21**, 167 (2017).

[58] K. T. Butler, D. W. Davies, H. Cartwright, O. Isayev and A. Walsh. *Machine learning for molecular and materials science.* Nature **559**, 547 (2018).

[59] D. Morgan and R. Jacobs. *Opportunities and Challenges for Machine Learning in Materials Science.* Annual Review of Materials Research **50** (2020).

[60] J. Wei, X. Chu, X.-Y. Sun, K. Xu, H.-X. Deng, J. Chen, Z. Wei and M. Lei. *Machine learning in materials science.* InfoMat **1**, 338 (2019).

[61] T. Mueller, A. G. Kusne and R. Ramprasad. *Machine learning in materials science: Recent progress and emerging applications.* Reviews in Computational Chemistry **29**, 186 (2016).

[62] L. Ward, R. Liu, A. Krishna, V. I. Hegde, A. Agrawal, A. Choudhary and C. Wolverton. *Including crystal structure attributes in machine learning models of formation energies via Voronoi tessellations.* Physical Review B **96**, 024104 (2017).

[63] L. Ward, A. Agrawal, A. Choudhary and C. Wolverton. *A general-purpose machine learning framework for predicting properties of inorganic materials.* npj Computational Materials **2**, 16028 (2016).

[64] M. Rupp, A. Tkatchenko, K.-R. Müller and O. A. Von Lilienfeld. *Fast and accurate modeling of molecular atomization energies with machine learning.* Physical Review Letters **108**, 058301 (2012).

[65] M. F. Langer, A. Goeßmann and M. Rupp. *Representations of molecules and materials for interpolation of quantum-mechanical simulations via machine learning.* arXiv preprint arXiv:2003.12081 (2020).

[66] M. Rupp. *Machine learning for quantum mechanics in a nutshell.* International Journal of Quantum Chemistry **115**, 1058 (2015).

[67] C. Sutton, M. Boley, L. M. Ghiringhelli, M. Rupp, J. Vreeken and M. Scheffler. *Identifying domains of applicability of machine learning models for materials science.* Nature Communications **11**, 1 (2020).

[68] T. B. Blank, S. D. Brown, A. W. Calhoun and D. J. Doren. *Neural network models of potential energy surfaces.* The Journal of Chemical Physics **103**, 4129 (1995).

[69] S. Lorenz, A. Groß and M. Scheffler. *Representing high-dimensional potential-energy surfaces for reactions at surfaces by neural networks.* Chemical Physics Letters **395**, 210 (2004).

[70] J. Behler and M. Parrinello. *Generalized neural-network representation of high-dimensional potential-energy surfaces.* Physical Review Letters **98**, 146401 (2007).

[71] J. Behler. *First principles neural network potentials for reactive simulations of large molecular and condensed systems.* Angewandte Chemie International Edition **56**, 12828 (2017).

[72] K. Schutt, P. Kessel, M. Gastegger, K. Nicoli, A. Tkatchenko and K.-R. Müller. *SchNetPack: A deep learning toolbox for atomistic systems.* Journal of Chemical Theory and Computation **15**, 448 (2018).

[73] G. Carleo and M. Troyer. *Solving the quantum many-body problem with artificial neural networks.* Science **355**, 602 (2017).

[74] M. Ruggeri, S. Moroni and M. Holzmann. *Nonlinear network description for many-body quantum systems in continuous space.* Physical Review Letters **120**, 205302 (2018).

[75] H. Saito. *Method to solve quantum few-body problems with artificial neural networks.* Journal of the Physical Society of Japan **87**, 074002 (2018).

[76] J. Han, L. Zhang and E. Weinan. *Solving many-electron Schrödinger equation using deep neural networks.* Journal of Computational Physics **399**, 108929 (2019).

[77] J. Hermann, Z. Schätzle and F. Noé. *Deep-neural-network solution of the electronic Schrödinger equation.* Nature Chemistry **12**, 891 (2020).

[78] D. Pfau, J. S. Spencer, A. G. Matthews and W. M. C. Foulkes. *Ab initio solution of the many-electron Schrödinger equation with deep neural networks.* Physical Review Research **2**, 033429 (2020).

[79] A. W. Senior, R. Evans, J. Jumper, J. Kirkpatrick, L. Sifre, T. Green, C. Qin, A. Žídek, A. W. Nelson, A. Bridgland et al. *Improved protein structure prediction using potentials from deep learning.* Nature **577**, 706 (2020).

[80] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard and L. D. Jackel. *Backpropagation applied to handwritten zip code recognition.* Neural Computation **1**, 541 (1989).

[81] M. Ziatdinov, O. Dyck, A. Maksov, X. Li, X. Sang, K. Xiao, R. R. Unocic, R. Vasudevan, S. Jesse and S. V. Kalinin. *Deep learning of atomically resolved scanning transmission electron microscopy images: chemical identification and tracking local transformations.* ACS Nano **11**, 12742 (2017).

[82] N. Borodinov, S. Neumayer, S. V. Kalinin, O. S. Ovchinnikova, R. K. Vasudevan and S. Jesse. *Deep neural networks for understanding noisy data applied to physical property extraction in scanning probe microscopy.* npj Computational Materials **5**, 1 (2019).

[83] R. K. Vasudevan, N. Laanait, E. M. Ferragut, K. Wang, D. B. Geohegan, K. Xiao, M. Ziatdinov, S. Jesse, O. Dyck and S. V. Kalinin. *Mapping mesoscopic phase evolution during E-beam induced transformations via deep learning of atomically resolved images.* npj Computational Materials **4**, 1 (2018).

[84] M. Ziatdinov, O. Dyck, X. Li, B. G. Sumpter, S. Jesse, R. K. Vasudevan and S. V. Kalinin. *Building and exploring libraries of atomic defects in graphene: Scanning*

transmission electron and scanning tunneling microscopy study. Science Advances **5**, eaaw8989 (2019).

[85] D. Jha, L. Ward, A. Paul, W.-k. Liao, A. Choudhary, C. Wolverton and A. Agrawal. *ElemNet: deep learning the chemistry of materials from only elemental composition.* Scientific Reports **8**, 17593 (2018).

[86] D. Jha, K. Choudhary, F. Tavazza, W.-k. Liao, A. Choudhary, C. Campbell and A. Agrawal. *Enhancing materials property prediction by leveraging computational and experimental data using deep transfer learning.* Nature Communications **10**, 1 (2019).

[87] H. Liang, V. Stanev, A. G. Kusne and I. Takeuchi. *CRYSPNet: Crystal structure predictions via neural networks.* Physical Review Materials **4**, 123802 (2020).

[88] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals and G. E. Dahl. *Neural message passing for quantum chemistry.* In *International Conference on Machine Learning*, 1263–1272 (PMLR, 2017).

[89] K. Schütt, P.-J. Kindermans, H. E. S. Felix, S. Chmiela, A. Tkatchenko and K.-R. Müller. *Schnet: A continuous-filter convolutional neural network for modeling quantum interactions.* In *Advances in neural information processing systems*, 991–1001 (2017).

[90] K. T. Schütt, H. E. Sauceda, P.-J. Kindermans, A. Tkatchenko and K.-R. Müller. *SchNet–A deep learning architecture for molecules and materials.* The Journal of Chemical Physics **148**, 241722 (2018).

[91] T. Xie and J. C. Grossman. *Hierarchical visualization of materials space with graph convolutional neural networks.* The Journal of Chemical Physics **149**, 174111 (2018).

[92] T. E. Smidt, M. Geiger and B. K. Miller. *Finding symmetry breaking order parameters with Euclidean neural networks.* Physical Review Research **3**, L012002 (2021).

[93] A. Gupta, A. T. Müller, B. J. Huisman, J. A. Fuchs, P. Schneider and G. Schneider. *Generative recurrent networks for de novo drug design.* Molecular Informatics **37**, 1700111 (2018).

[94] F. Noé, S. Olsson, J. Köhler and H. Wu. *Boltzmann generators: Sampling equilibrium states of many-body systems with deep learning.* Science **365**, eaaw1147 (2019).

[95] Y. B. Varolgüneş, T. Bereau and J. F. Rudzinski. *Interpretable embeddings from molecular simulations using Gaussian mixture variational autoencoders.* Machine Learning: Science and Technology **1**, 015012 (2020).

[96] N. Baddoo. *Stainless steel in construction: A review of research, applications, challenges and opportunities.* Journal of Constructional Steel Research **64**, 1199 (2008).

[97] M. Herbig, D. Raabe, Y. Li, P. Choi, S. Zaefferer and S. Goto. *Atomic-scale quantification of grain boundary segregation in nanocrystalline material.* Physical Review Letters **112**, 126103 (2014).

[98] T. Meiners, T. Frolov, R. E. Rudd, G. Dehm and C. H. Liebscher. *Observations of grain-boundary phase transformations in an elemental metal.* Nature **579**, 375 (2020).

[99] A. C. Ferrari, F. Bonaccorso, V. Fal'Ko, K. S. Novoselov, S. Roche, P. Bøggild, S. Borini, F. H. Koppens, V. Palermo, N. Pugno et al. *Science and technology roadmap for graphene, related two-dimensional crystals, and hybrid systems.* Nanoscale **7**, 4598 (2015).

[100] M. F. De Volder, S. H. Tawfick, R. H. Baughman and A. J. Hart. *Carbon nanotubes: present and future commercial applications.* Science **339**, 535 (2013).

[101] M. J. Mehl, D. Hicks, C. Toher, O. Levy, R. M. Hanson, G. Hart and S. Curtarolo. *The AFLOW library of crystallographic prototypes: part 1.* Computational Materials Science **136**, S1 (2017).

[102] D. Hicks, M. J. Mehl, E. Gossett, C. Toher, O. Levy, R. M. Hanson, G. Hart and S. Curtarolo. *The AFLOW Library of Crystallographic Prototypes: part 2.* Computational Materials Science **161**, S1 (2019).

[103] J. E. Saal, S. Kirklin, M. Aykol, B. Meredig and C. Wolverton. *Materials design and discovery with high-throughput density functional theory: the open quantum materials database (OQMD).* JOM **65**, 1501 (2013).

[104] A. Jain, G. Hautier, C. J. Moore, S. P. Ong, C. C. Fischer, T. Mueller, K. A. Persson and G. Ceder. *A high-throughput infrastructure for density-functional theory calculations.* Computational Materials Science **50**, 2295 (2011).

[105] 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 et al. *The Computational 2D Materials Database: high-throughput modeling and discovery of atomically thin crystals.* 2D Materials **5**, 042002 (2018).

[106] N. Mounet, M. Gibertini, P. Schwaller, D. Campi, A. Merkys, A. Marrazzo, T. Sohier, I. E. Castelli, A. Cepellotti, G. Pizzi et al. *Two-dimensional materials from high-throughput computational exfoliation of experimentally known compounds.* Nature Nanotechnology **13**, 246 (2018).

[107] S. J. Pennycook and P. D. Nellist. *Scanning Transmission Electron Microscopy: Imaging and Analysis* (Springer New York, 2011).

[108] B. Gault, M. P. Moody, J. M. Cairney and S. P. Ringer. *Atom probe crystallography.* Materials Today **15**, 378 (2012).

[109] J. Zhou, Y. Yang, P. Ercius and J. Miao. *Atomic electron tomography in three and four dimensions.* MRS Bulletin **45**, 290 (2020).

[110] B. Gault. *A brief overview of atom probe tomography research.* Applied Microscopy **46**, 117 (2016).

[111] A. Togo and I. Tanaka. *Spglib: a software library for crystal symmetry search.* arXiv preprint arXiv:1808.01590 (2018).

[112] D. Hicks, C. Oses, E. Gossett, G. Gomez, R. H. Taylor, C. Toher, M. J. Mehl, O. Levy and S. Curtarolo. *AFLOW-SYM: platform for the complete, automatic and self-consistent symmetry analysis of crystals.* Acta Crystallographica Section A: Foundations and Advances **74**, 184 (2018).

[113] J. D. Honeycutt and H. C. Andersen. *Molecular dynamics study of melting and freezing of small Lennard-Jones clusters.* Journal of Physical Chemistry **91**, 4950 (1987).

[114] A. Stukowski. *Structure identification methods for atomistic simulations of crystalline materials.* Modelling and Simulation in Materials Science and Engineering **20**, 045021 (2012).

[115] G. Ackland and A. Jones. *Applications of local crystal structure measures in experiment and simulation.* Physical Review B **73**, 054104 (2006).

[116] P. M. Larsen, S. Schmidt and J. Schiøtz. *Robust structural identification via polyhedral template matching.* Modelling and Simulation in Materials Science and Engineering **24**, 055007 (2016).

[117] H. W. Lin, M. Tegmark and D. Rolnick. *Why does deep and cheap learning work so well?* Journal of Statistical Physics **168**, 1223 (2017).

[118] S. Hong, K.-i. Nomura, A. Krishnamoorthy, P. Rajak, C. Sheng, R. K. Kalia, A. Nakano and P. D. Vashishta. *Defect Healing in Layered Materials: A Machine Learning-Assisted Characterization of MoS2 Crystal-Phases.* The Journal of Physical Chemistry Letters (2019).

[119] P. Geiger and C. Dellago. *Neural networks for local structure detection in polymorphic systems.* The Journal of Chemical Physics **139**, 164105 (2013).

[120] W. F. Reinhart, A. W. Long, M. P. Howard, A. L. Ferguson and A. Z. Panagiotopoulos. *Machine learning for autonomous crystal structure identification.* Soft Matter **13**, 4733 (2017).

[121] C. Dietz, T. Kretz and M. Thoma. *Machine-learning approach for local classification of crystalline structures in multiphase systems.* Physical Review E **96**, 011301 (2017).

[122] A. Leitherer, A. Ziletti and L. M. Ghiringhelli. *Robust recognition and exploratory analysis of crystal structures via Bayesian deep learning.* Nature Communications **12**, 6234 (2021).

[123] Y. Gal and Z. Ghahramani. *Dropout as a Bayesian approximation: Representing model uncertainty in deep learning.* In *International Conference on Machine Learning*, 1050–1059 (2016).

[124] Y. Gal. *Uncertainty in deep learning.* Ph.D. thesis, University of Cambridge (2016).

[125] J. Behler. *Atom-centered symmetry functions for constructing high-dimensional neural network potentials.* The Journal of Chemical Physics **134**, 074106 (2011).

[126] A. P. Bartók, M. C. Payne, R. Kondor and G. Csányi. *Gaussian approximation potentials: The accuracy of quantum mechanics, without the electrons.* Physical Review Letters **104**, 136403 (2010).

[127] A. P. Bartók, R. Kondor and G. Csányi. *On representing chemical environments.* Physical Review B **87**, 184115 (2013).

[128] H. Huo and M. Rupp. *Unified representation for machine learning of molecules and crystals.* arXiv preprint arXiv:1704.06439 (2017).

[129] A. V. Shapeev. *Moment tensor potentials: A class of systematically improvable interatomic potentials.* Multiscale Modeling & Simulation **14**, 1153 (2016).

[130] A. P. Bartók and G. Csányi. *Gaussian approximation potentials: A brief tutorial introduction.* International Journal of Quantum Chemistry **115**, 1051 (2015).

[131] S. De, A. P. Bartók, G. Csányi and M. Ceriotti. *Comparing molecules and solids across structural and alchemical space.* Physical Chemistry Chemical Physics **18**, 13754 (2016).

[132] C. Nyshadham, M. Rupp, B. Bekker, A. V. Shapeev, T. Mueller, C. W. Rosenbrock, G. Csányi, D. W. Wingate and G. L. Hart. *Machine-learned multi-system surrogate models for materials prediction.* npj Computational Materials **5**, 1 (2019).

[133] C. W. Rosenbrock, E. R. Homer, G. Csányi and G. L. Hart. *Discovering the building blocks of atomic systems using machine learning: application to grain boundaries.* npj Computational Materials **3**, 1 (2017).

[134] A. Zhang, Z. C. Lipton, M. Li and A. J. Smola. *Dive into Deep Learning.* arXiv preprint arXiv:2106.11342 (2021).

[135] F. Rosenblatt. *The perceptron, a perceiving and recognizing automaton (Project Para)* (Cornell Aeronautical Laboratory, 1957).

[136] S. Hochreiter and J. Schmidhuber. *Long short-term memory.* Neural Computation **9**, 1735 (1997).

[137] K. He, X. Zhang, S. Ren and J. Sun. *Deep residual learning for image recognition.* In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 770–778 (2016).

[138] Y. Bengio, A. Courville and P. Vincent. *Representation learning: A review and new perspectives.* IEEE Transactions on Pattern Analysis and Machine Intelligence **35**, 1798 (2013).

[139] G. Cybenko. *Approximation by superpositions of a sigmoidal function.* Mathematics of Control, Signals and Systems **2**, 303 (1989).

[140] K. Hornik, M. Stinchcombe and H. White. *Multilayer feedforward networks are universal approximators.* Neural Networks **2**, 359 (1989).

[141] M. Leshno, V. Y. Lin, A. Pinkus and S. Schocken. *Multilayer feedforward networks with a nonpolynomial activation function can approximate any function.* Neural Networks **6**, 861 (1993).

[142] D. R. Wilson and T. R. Martinez. *The general inefficiency of batch training for gradient descent learning.* Neural Networks **16**, 1429 (2003).

[143] B. T. Polyak. *Some methods of speeding up the convergence of iteration methods.* USSR Computational Mathematics and Mathematical Physics **4**, 1 (1964).

[144] G. Hinton, N. Srivastava and K. Swersky. *Neural networks for machine learning lecture 6a overview of mini-batch gradient descent.* Coursera video lecture (2012).

[145] R. A. Jacobs. *Increased rates of convergence through learning rate adaptation.* Neural Networks **1**, 295 (1988).

[146] J. Duchi, E. Hazan and Y. Singer. *Adaptive subgradient methods for online learning and stochastic optimization.* Journal of Machine Learning Research **12** (2011).

[147] S. Reddi, M. Zaheer, D. Sachan, S. Kale and S. Kumar. *Adaptive methods for nonconvex optimization.* In *Proceeding of 32nd Conference on Neural Information Processing Systems* (2018).

[148] Q. V. Le, J. Ngiam, A. Coates, A. Lahiri, B. Prochnow and A. Y. Ng. *On optimization methods for deep learning.* In *ICML* (2011).

[149] X. Glorot and Y. Bengio. *Understanding the difficulty of training deep feedforward neural networks.* In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, 249–256 (JMLR Workshop and Conference Proceedings, 2010).

[150] R. Tibshirani. *Regression shrinkage and selection via the lasso.* Journal of the Royal Statistical Society: Series B (Methodological) **58**, 267 (1996).

[151] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever and R. R. Salakhutdinov. *Improving neural networks by preventing co-adaptation of feature detectors.* arXiv preprint arXiv:1207.0580 (2012).

[152] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever and R. Salakhutdinov. *Dropout: a simple way to prevent neural networks from overfitting.* The Journal of Machine Learning Research **15**, 1929 (2014).

[153] C. M. Bishop. *Regularization and complexity control in feed-forward networks.* International Conference on Artificial Neural Networks (1995).

[154] J. Sjöberg and L. Ljung. *Overtraining, regularization and searching for a minimum, with application to neural networks.* International Journal of Control **62**, 1391 (1995).

[155] J. Bergstra, D. Yamins and D. D. Cox. *Making a Science of Model Search: Hyperparameter Optimization in Hundreds of Dimensions for Vision Architectures.* In *Proceedings of the 30th International Conference on Machine Learning*, ICML'13, I–115–I–123 (JMLR.org, 2013).

[156] J. Bergstra and Y. Bengio. *Random search for hyper-parameter optimization.* Journal of Machine Learning Research **13** (2012).

[157] E. Brochu, V. M. Cora and N. De Freitas. *A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning.* arXiv preprint arXiv:1012.2599 (2010).

[158] Z. Ghahramani. *Probabilistic machine learning and artificial intelligence.* Nature **521**, 452 (2015).

[159] P. I. Frazier. *A tutorial on Bayesian optimization.* arXiv preprint arXiv:1807.02811 (2018).

[160] C. E. Rasmussen. *Gaussian processes in machine learning.* In *Summer School on Machine Learning*, 63–71 (Springer, 2003).

[161] F. Hutter. *Automated configuration of algorithms for solving hard computational problems.* Ph.D. thesis, University of British Columbia (2009).

[162] D. R. Jones. *A taxonomy of global optimization methods based on response surfaces.* Journal of Global Optimization **21**, 345 (2001).

[163] J. S. Bergstra, R. Bardenet, Y. Bengio and B. Kégl. *Algorithms for hyper-parameter optimization.* In *Advances in Neural Information Processing Systems*, 2546–2554 (2011).

[164] J. Bergstra, D. Yamins, D. D. Cox et al. *Hyperopt: A Python library for optimizing the hyperparameters of machine learning algorithms.* In *Proceedings of the 12th Python in Science Conference*, volume 13, 20 (Citeseer, 2013).

[165] L. Smith and Y. Gal. *Understanding measures of uncertainty for adversarial example detection.* arXiv preprint arXiv:1803.08533 (2018).

[166] A. Kendall and Y. Gal. *What uncertainties do we need in bayesian deep learning for computer vision?* arXiv preprint arXiv:1703.04977 (2017).

[167] M. I. Jordan, Z. Ghahramani, T. S. Jaakkola and L. K. Saul. *An introduction to variational methods for graphical models.* Machine Learning **37**, 183 (1999).

[168] N. Houlsby, F. Huszár, Z. Ghahramani and M. Lengyel. *Bayesian active learning for classification and preference learning.* arXiv preprint arXiv:1112.5745 (2011).

[169] Z. Ghahramani. *Unsupervised learning.* In *Summer School on Machine Learning*, 72–112 (Springer, 2003).

[170] C. Sammut and G. I. Webb. *Encyclopedia of machine learning* (Springer Science & Business Media, 2011).

[171] H.-P. Kriegel, P. Kröger, J. Sander and A. Zimek. *Density-based clustering.* Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery **1**, 231 (2011).

[172] J. MacQueen et al. *Some methods for classification and analysis of multivariate observations.* In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, 281–297 (Oakland, CA, USA, 1967).

[173] D. M. Hawkins. *Identification of outliers*, volume 11 (Springer, 1980).

[174] R. J. Campello, D. Moulavi and J. Sander. *Density-based clustering based on hierarchical density estimates.* In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, 160–172 (Springer, 2013).

[175] M. Ester, H.-P. Kriegel, J. Sander, X. Xu et al. *A density-based algorithm for discovering clusters in large spatial databases with noise.* KDD Proceedings **96**, 226 (1996).

[176] E. Schubert, J. Sander, M. Ester, H. P. Kriegel and X. Xu. *DBSCAN revisited, revisited: why and how you should (still) use DBSCAN.* ACM Transactions on Database Systems **42**, 1 (2017).

[177] L. McInnes and J. Healy. *Accelerated hierarchical density based clustering.* In *2017 IEEE International Conference on Data Mining Workshops (ICDMW)*, 33–42 (IEEE, 2017).

[178] L. McInnes, J. Healy and S. Astels. *HDBSCAN: Hierarchical density based clustering.* The Journal of Open Source Software **2**, 205 (2017).

[179] S. C. Johnson. *Hierarchical clustering schemes.* Psychometrika **32**, 241 (1967).

[180] F. Murtagh and P. Contreras. *Algorithms for hierarchical clustering: an overview.* Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery **2**, 86 (2012).

[181] J. Eldridge, M. Belkin and Y. Wang. *Beyond hartigan consistency: Merge distortion metric for hierarchical clustering.* In *Conference on Learning Theory*, 588–606 (PMLR, 2015).

[182] K. Pearson. *LIII. On lines and planes of closest fit to systems of points in space.* The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science **2**, 559 (1901).

[183] H. Hotelling. *Analysis of a complex of statistical variables into principal components.* Journal of Educational Psychology **24**, 417 (1933).

[184] L. Van der Maaten and G. Hinton. *Visualizing data using t-SNE.* Journal of Machine Learning Research **9** (2008).

[185] L. McInnes, J. Healy and J. Melville. *UMAP: Uniform manifold approximation and projection for dimension reduction.* arXiv preprint arXiv:1802.03426 (2018).

[186] L. Cayton. *Algorithms for manifold learning.* Technical Report CS2008-0923, UCSD (2005).

[187] J. P. May. *Simplicial objects in algebraic topology*, volume 11 (University of Chicago Press, 1992).

[188] S. Mac Lane. *Categories for the working mathematician*, volume 5 (Springer Science & Business Media, 2013).

[189] A. Coenen and A. Pearce. https://github.com/PAIR-code/understanding-umap (2019).

[190] M. Wattenberg, F. Viégas and I. Johnson. *How to Use t-SNE Effectively.* Distill (2016).

[191] K. R. Moon, D. van Dijk, Z. Wang, S. Gigante, D. B. Burkhardt, W. S. Chen, K. Yim, A. van den Elzen, M. J. Hirn, R. R. Coifman et al. *Visualizing structure and transitions in high-dimensional biological data.* Nature Biotechnology **37**, 1482 (2019).

[192] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu and X. Zheng. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems* (2015).

[193] F. Chollet. *Keras.* https://keras.io (2015).

[194] L. Himanen, M. O. Jäger, E. V. Morooka, F. F. Canova, Y. S. Ranawat, D. Z. Gao, P. Rinke and A. S. Foster. *DScribe: Library of descriptors for machine learning in materials science.* Computer Physics Communications **247**, 106949 (2020).

[195] J. Mavračić, F. C. Mocanu, V. L. Deringer, G. Csányi and S. R. Elliott. *Similarity between amorphous and crystalline phases: The case of TiO2.* The Journal of Physical Chemistry Letters **9**, 2985 (2018).

[196] J. Patterson and A. Gibson. *Deep Learning: A Practitioner's Approach* (O'Reilly Media, Inc., 2017).

[197] A. Canziani, A. Paszke and E. Culurciello. *An analysis of deep neural network models for practical applications.* arXiv preprint arXiv:1605.07678 (2016).

[198] A. Shrestha and A. Mahmood. *Review of Deep Learning Algorithms and Architectures.* IEEE Access **7**, 53040 (2019).

[199] N. W. Ashcroft and N. D. Mermin. *Solid State Physics* (Cengage Learning, London, 2011).

[200] D. G. Pettifor. *Bonding and Structure of Molecules and Solids* (Oxford University Press, 1995).

[201] R. C. Reed. *The Superalloys: Fundamentals and Applications* (Cambridge University Press, 2008).

[202] I. E. Castelli, K. S. Thygesen and K. W. Jacobsen. *Calculated optical absorption of different perovskite phases.* Journal of Materials Chemistry A **3**, 12343 (2015).

[203] M. Pandey and K. W. Jacobsen. *Promising quaternary chalcogenides as high-band-gap semiconductors for tandem photoelectrochemical water splitting devices: A computational screening approach.* Physical Review Materials **2**, 105402 (2018).

[204] K. Novoselov, A. Mishchenko, A. Carvalho and A. C. Neto. *2D materials and van der Waals heterostructures.* Science **353**, aac9439 (2016).

[205] D. D. Landis, J. S. Hummelshoj, S. Nestorov, J. Greeley, M. Dulak, T. Bligaard, J. K. Norskov and K. W. Jacobsen. *The computational materials repository.* Computing in Science & Engineering **14**, 51 (2012).

[206] A. H. Larsen, J. J. Mortensen, J. Blomqvist, I. E. Castelli, R. Christensen, M. Dułak, J. Friis, M. N. Groves, B. Hammer, C. Hargus et al. *The atomic simulation environment—a Python library for working with atoms.* Journal of Physics: Condensed Matter **29**, 273002 (2017).

[207] A. Stukowski. *Visualization and analysis of atomistic simulation data with OVITO–the Open Visualization Tool.* Modelling and Simulation in Materials Science and Engineering **18**, 015012 (2009).

[208] J. Long, E. Shelhamer and T. Darrell. *Fully convolutional networks for semantic segmentation.* In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 3431–3440 (2015).

[209] E. C. Bain and N. Y. Dunkirk. *The nature of martensite.* Trans. AIME **70**, 25 (1924).

[210] J.-C. Zhao and M. R. Notis. *Continuous cooling transformation kinetics versus isothermal transformation kinetics of steels: a phenomenological rationalization of experimental observations.* Materials Science and Engineering: R: Reports **15**, 135 (1995).

[211] G. Grimvall, B. Magyari-Köpe, V. Ozoliņš and K. A. Persson. *Lattice instabilities in metallic elements.* Reviews of Modern Physics **84**, 945 (2012).

[212] P. Alippi, P. M. Marcus and M. Scheffler. *Strained tetragonal states and bain paths in metals.* Physical Review Letters **78**, 3892 (1997).

[213] J. Buschbeck, I. Opahle, M. Richter, U. K. Rößler, P. Klaer, M. Kallmayer, H. J. Elmers, G. Jakob, L. Schultz and S. Fähler. *Full Tunability of Strain along the fcc-bcc Bain Path in Epitaxial Films and Consequences for Magnetic Properties.* Physical Review Letters **103** (2009).

[214] M. I. Haftel and K. Gall. *Density functional theory investigation of surface-stress-induced phase transformations in fcc metal nanowires.* Physical Review B - Condensed Matter and Materials Physics **74** (2006).

[215] W. P. Davey. *The lattice parameter and density of pure tungsten.* Physical Review **26**, 736 (1925).

[216] M. Straumanis and L. Yu. *Lattice parameters, densities, expansion coefficients and perfection of structure of Cu and of Cu–In α phase.* Acta Crystallographica Section A: Crystal Physics, Diffraction, Theoretical and General Crystallography **25**, 676 (1969).

[217] V. Deshpande and R. Pawar. *Anisotropic thermal expansion of indium.* Acta Crystallographica Section A: Crystal Physics, Diffraction, Theoretical and General Crystallography **25**, 415 (1969).

[218] W. Zachariasen. *On the crystal structure of protactinium metal.* Acta Crystallographica **12**, 698 (1959).

[219] O. Isayev, D. Fourches, E. N. Muratov, C. Oses, K. Rasch, A. Tropsha and S. Curtarolo. *Materials cartography: representing and mining materials space using structural and electronic fingerprints.* Chemistry of Materials **27**, 735 (2015).

[220] C. W. Glass, A. R. Oganov and N. Hansen. *USPEX—Evolutionary crystal structure prediction.* Computer Physics Communications **175**, 713 (2006).

[221] P. Hirel. *Atomsk: a tool for manipulating and converting atomic data files.* Computer Physics Communications **197**, 212 (2015).

[222] L. M. Ghiringhelli. *Interpretability of machine-learning models in physical sciences.* arXiv preprint arXiv:2104.10443 (2021).

[223] A. B. Parsa, P. Wollgramm, H. Buck, C. Somsen, A. Kostka, I. Povstugar, P. P. Choi, D. Raabe, A. Dlouhy, J. Müller, E. Spiecker, K. Demtroder, J. Schreuer, K. Neuking and G. Eggeler. *Advanced scale bridging microstructure analysis of single crystal Ni-base superalloys.* Advanced Engineering Materials **17**, 216 (2015).

[224] M. Ziatdinov, A. Ghosh, T. Wong and S. V. Kalinin. *AtomAI: A Deep Learning Framework for Analysis of Image and Spectroscopy Data in (Scanning) Transmission Electron Microscopy and Beyond.* arXiv preprint arXiv:2105.07485 (2021).

[225] M. Nord, P. E. Vullum, I. MacLaren, T. Tybell and R. Holmestad. *Atomap: a new software tool for the automated analysis of atomic resolution images using two-dimensional Gaussian fitting.* Advanced Structural and Chemical Imaging **3**, 1 (2017).

[226] D. Levine and P. J. Steinhardt. *Quasicrystals: a new class of ordered structures.* Physical Review Letters **53**, 2477 (1984).

[227] R. Li, Z. Li, Z. Dong and K. A. Khor. *A review of transmission electron microscopy of quasicrystals—how are atoms arranged?* Crystals **6**, 105 (2016).

[228] M. Scott, C.-C. Chen, M. Mecklenburg, C. Zhu, R. Xu, P. Ercius, U. Dahmen, B. Regan and J. Miao. *Electron tomography at 2.4-ångström resolution.* Nature **483**, 444 (2012).

[229] J. Miao, P. Ercius and S. J. Billinge. *Atomic electron tomography: 3D structures without crystals.* Science **353**, aaf2157 (2016).

[230] C.-C. Chen, C. Zhu, E. R. White, C.-Y. Chiu, M. Scott, B. Regan, L. D. Marks, Y. Huang and J. Miao. *Three-dimensional imaging of dislocations in a nanoparticle at atomic resolution.* Nature **496**, 74 (2013).

[231] R. Xu, C.-C. Chen, L. Wu, M. Scott, W. Theis, C. Ophus, M. Bartels, Y. Yang, H. Ramezani-Dakhel, M. R. Sawaya et al. *Three-dimensional coordinates of individual atoms in materials revealed by electron tomography.* Nature Materials **14**, 1099 (2015).

[232] J. Zhou, Y. Yang, Y. Yang, D. S. Kim, A. Yuan, X. Tian, C. Ophus, F. Sun, A. K. Schmid, M. Nathanson et al. *Observing crystal nucleation in four dimensions using atomic electron tomography.* Nature **570**, 500 (2019).

[233] X. Tian, D. S. Kim, S. Yang, C. J. Ciccarino, Y. Gong, Y. Yang, Y. Yang, B. Duschatko, Y. Yuan, P. M. Ajayan et al. *Correlating the three-dimensional atomic defects and electronic properties of two-dimensional transition metal dichalcogenides.* Nature Materials **19**, 867 (2020).

[234] S. Sun. *Recent advances in chemical synthesis, self-assembly, and applications of FePt nanoparticles.* Advanced Materials **18**, 393 (2006).

[235] Y. Yang, J. Zhou, F. Zhu, Y. Yuan, D. J. Chang, D. S. Kim, M. Pham, A. Rana, X. Tian, Y. Yao et al. *Determining the three-dimensional atomic structure of an amorphous solid.* Nature **592**, 60 (2021).

[236] Y. Yuan, D. S. Kim, J. Zhou, D. J. Chang, F. Zhu, Y. Nagaoka, Y. Yang, M. Pham, S. J. Osher, O. Chen et al. *Three-dimensional atomic packing in amorphous solids with liquid-like structure.* Nature Materials 1–8 (2021).

[237] Y. Yang, C.-C. Chen, M. Scott, C. Ophus, R. Xu, A. Pryor, L. Wu, F. Sun, W. Theis, J. Zhou et al. *Deciphering chemical order/disorder and material properties at the single-atom level.* Nature **542**, 75 (2017).

[238] S. Curtarolo, D. Morgan, K. Persson, J. Rodgers and G. Ceder. *Predicting crystal structures with data mining of quantum calculations.* Physical Review Letters **91**, 135503 (2003).

[239] C. C. Fischer, K. J. Tibbetts, D. Morgan and G. Ceder. *Predicting crystal structure by merging data mining with quantum mechanics.* Nature Materials **5**, 641 (2006).

[240] Y. Liu, M. O'Keeffe, M. M. Treacy and O. M. Yaghi. *The geometry of periodic knots, polycatenanes and weaving from a chemical perspective: a library for reticular chemistry.* Chemical Society Reviews **47**, 4642 (2018).

[241] Y. Gal, R. Islam and Z. Ghahramani. *Deep Bayesian active learning with image data.* In *Proceedings of the 34th International Conference on Machine Learning*, volume 70, 1183–1192 (2017).

[242] P. M. Larsen, E. L. Pang, P. A. Parrilo and K. W. Jacobsen. *Minimum-strain symmetrization of Bravais lattices.* Physical Review Research **2**, 013077 (2020).

[243] L. Himanen, P. Rinke and A. S. Foster. *Materials structure genealogy and high-throughput topological classification of surfaces and 2D materials.* npj Computational Materials **4**, 1 (2018).

[244] J. J. Möller and E. Bitzek. *BDA: A novel method for identifying defects in body-centered cubic crystals.* MethodsX **3**, 279 (2016).

[245] B. Gault, M. P. Moody, F. De Geuser, A. La Fontaine, L. T. Stephenson, D. Haley and S. P. Ringer. *Spatial resolution in atom probe tomography.* Microscopy and Microanalysis **16**, 99 (2010).

[246] V. J. Araullo-Peters, A. Breen, A. V. Ceguerra, B. Gault, S. P. Ringer and J. M. Cairney. *A new systematic framework for crystallographic analysis of atom probe data.* Ultramicroscopy **154**, 7 (2015).

[247] C. Oberdorfer, S. M. Eich and G. Schmitz. *A full-scale simulation approach for atom probe tomography.* Ultramicroscopy **128**, 55 (2013).

[248] M. Kühbach, A. Breen, M. Herbig and B. Gault. *Building a Library of Simulated Atom Probe Data for Different Crystal Structures and Tip Orientations Using TAPSim.* Microscopy and Microanalysis **25**, 320 (2019).

[249] M. Kühbach, P. Bajaj, M. H. Celik, E. A. Jägle and B. Gault. *On Strong Scaling and Open Source Tools for Analyzing Atom Probe Tomography Data.* arXiv preprint arXiv:2004.05188 (2020).

[250] A. Prakash, J. Guénolé, J. Wang, J. Müller, E. Spiecker, M. J. Mills, I. Povstugar, P. Choi, D. Raabe and E. Bitzek. *Atom probe informed simulations of dislocation–precipitate interactions reveal the importance of local interface curvature.* Acta Materialia **92**, 33 (2015).

[251] A. Prakash, M. Hummel, S. Schmauder and E. Bitzek. *Nanosculpt: A methodology for generating complex realistic configurations for atomistic simulations.* MethodsX **3**, 219 (2016).

[252] Q. Zhu, A. Samanta, B. Li, R. E. Rudd and T. Frolov. *Predicting phase behavior of grain boundaries with evolutionary search and machine learning.* Nature Communications **9**, 1 (2018).

Selbstständigkeitserklärung

Ich erkläre, dass ich die Dissertation selbstständig und nur unter Verwendung der von mir gemäß §7 Abs. 3 der Promotionsordnung der Mathematisch-Naturwissenschaftlichen Fakultät, veröffentlicht im Amtlichen Mitteilungsblatt der Humboldt-Universität zu Berlin Nr. 42/2018 am 11.07.2018 angegebenen Hilfsmittel angefertigt habe.

Ort, Datum:

Unterschrift: