Karlsruhe Institute of Technology

Applications of Machine Learning in Materials Science

Zur Erlangung des akademischen Grades eines
DOKTORS DER INGENIEURWISSENSCHAFTEN (Dr.-Ing.)
von der KIT-Fakultät für Maschinenbau des
Karlsruher Instituts für Technologie (KIT)

angenommene

DISSERTATION

von

M.Sc. Yinghan Zhao

Tag der mündlichen Prüfung:	15.04.2024
Hauptreferentin:	Prof. Dr. rer. nat. Britta Nestler
Korreferent:	Prof. Dr. Markus Reischl

Abstract

The increasing use of data-driven techniques in materials science, including machine learning, has given rise to materials informatics and shows great potential for expediting the discovery and development of advanced materials. Machine learning is currently widely explored in the materials science community, and its applications include discovering and screening novel materials candidates, accelerating expensive computational simulations, and enhancing the analysis of large amounts of complicated characterization data. This work aims to summarize fundamental concepts and introduce a unified workflow for utilizing machine learning in materials science. To illustrate its advantages and effectiveness, three studies are conducted to serve as practical examples of use, covering the most common aspects involved in the process of materials research: experiments, characterization, and simulation. For experiments, this work utilizes an optimization algorithm to determine the experimental parameters for the sintering synthesis process of an electrolyte material in the solid-state battery, with the goal of reducing the time needed to find the optimal experimental parameters. Similarly, this methodology is extended to find the most appropriate parameters for generating simulated digital porous membrane microstructures. For simulation, an (artificial) neural network-based model is utilized to fit a fast predictive model to build process-structure-property relationships for porous membrane microstructures. In particular, the representation learning ability of the model is exploited to automatically extract structural information. For characterization, this work explores different machine learning methods for the analysis of time-of-flight secondary ion mass spectrometry data measured from lithium compounds. Not only can a fast classification be built for identifying compositions, but interpretable chemical information can also be retrieved based on the feature importance of the model. These studies are typical examples that illustrate the benefits of applying machine learning methods in material research and the ideas presented can be extended to different scenarios. Furthermore, the experiences and reflections gained from these studies are discussed, along with suggestions for future research in this area.

Zusammenfassung

Der zunehmende Einsatz datengesteuerter Verfahren in der Materialwissenschaft, einschließlich des maschinellen Lernens, hat zur Werkstoffinformatik geführt und birgt ein großes Potenzial für die schnellere Entdeckung und Entwicklung fortschrittlicher Materialien. Das maschinelle Lernen wird derzeit in der Materialwissenschaft intensiv erforscht und zu seinen Anwendungen gehören die Entdeckung und das Screening neuer Materialien, die Beschleunigung teurer Computersimulationen und die Verbesserung der Analyse großer Mengen komplexer Charakterisierungsdaten. Das Ziel dieser Arbeit besteht darin, grundlegende Konzepte zusammenzufassen und einen einheitlichen Arbeitsablauf für den Einsatz von maschinellem Lernen in der Materialwissenschaft zu präsentieren. Um die Vorteile und Effektivität des maschinellen Lernens zu veranschaulichen, werden drei Studien durchgeführt. Diese dienen als praktische Anwendungsbeispiele und decken die häufigsten Aspekte im Prozess der Materialforschung ab: Experimente, Charakterisierung und Simulation. Für die Experimente wird in dieser Arbeit ein Optimierungsalgorithmus verwendet, um die experimentellen Parameter für den Sintersyntheseprozess eines Elektrolytmaterials in einer Festkörperbatterie zu bestimmen, mit dem Ziel, die Zeit zu reduzieren, die benötigt wird, um die optimalen experimentellen Parameter zu finden. Diese Methodik wird auch erweitert, um die am besten geeigneten Parameter für die Erzeugung simulierter digitaler poröser Mikrostrukturen zu finden. Für die Simulation wird ein auf einem (künstlichen) neuronalen Netzwerk basierendes Modell verwendet, um schnell Prozess-Struktur-Eigenschafts-Beziehungen für poröse Membranmikrostrukturen vorhersagen zu können. Das Modell nutzt insbesondere seine Fähigkeit, Repräsentationen zu lernen, um automatisch strukturelle Informationen zu extrahieren. Zur Charakterisierung werden in dieser Arbeit verschiedene Methoden des maschinellen Lernens zur Analyse von Flugzeit-Sekundärionen-Massenspektrometriedaten von Lithiumverbindungen untersucht. Es ist möglich, nicht nur eine schnelle Klassifizierung zur Identifizierung von Zusammensetzungen zu erstellen, sondern auch interpretierbare chemische Informationen auf der Grundlage der Bedeutung der Merkmale des Modells zu erfassen. Diese Studien sind typische Beispiele, die den Nutzen der Anwendung von Methoden des maschinellen Lernens in der Materialforschung verdeutlichen, und die vorgestellten Ideen können auf andere Szenarien erweitert werden. Darüber hinaus werden die Erfahrungen und Überlegungen, die bei diesen Studien gewonnen wurden, erörtert und Anmerkungen für die künftige Forschung in diesem Bereich gemacht.

Acknowledgments

This thesis stands as a testament to the collective efforts and contributions of many, and I am sincerely grateful for the privilege of learning and growing within this academic community. I am grateful to the Institute for Applied Materials - Microstructure Modeling and Simulation (IAM-MMS) at the Karlsruhe Institute of Technology (KIT) for providing the resources and academic environment essential for conducting this research. I am also thankful to the Federal Ministry of Education and Research (BMBF) for financially supporting my academic endeavors, which were crucial for the completion of this thesis.

First of all, I would like to express my deepest gratitude to Prof. Dr. Britta Nestler for her supervision, unwavering guidance, and invaluable suggestions throughout the entire journey of this research. I would also like to express my sincere appreciation to Prof. Dr. Markus Reischl for his willingness to be the co-supervisor of this thesis. His professional insights in the field have greatly inspired and benefited this research.

I am deeply grateful to my group leaders Dr. Michael Selzer and Dr. Arnd Koeppe for their constructive feedback, continuous support, and patient assistance at every stage of the research project. Their expertise has significantly enriched the quality of this work.

I would like to offer my special thanks to all the members of the "Kadi4Mat Team", in particular N. Brandt, P. Zschumme, J. Santoki, G. Tosato, L. Griem, P. Altschuh, J. Grolig, D. Rajagopal, Y. Ji, S. Jain, but also a large number of colleagues not listed for numerous discussions and cherished time spent together in the lab. I would also like to thank my colleagues F. Wang, Y. Wu, H. Zhang, and Y. Cai for a pleasant working atmosphere and varied conversations during the lunch breaks. For the technical and administrative support, I would like to thank D. Frank, C. Ratz, I. Heise, B. Hardt, M. Ritzinger, and L. Geisen.

I would also want to express my sincere appreciation to my collaborative partners, S. Otto, T. Lombardo, A. Henss, J. Janek, N. Schiffmann, E. Bucharsky, and K. Schell, who provided valuable ideas and professional support in the course of this work. I would also like to thank all the students who worked with me as research assistants or as part of their final theses.

Finally, I owe a debt of gratitude to my family, especially my parents, my grandmother, and my wife for their encouragement and understanding during this doctoral journey. Their love, support, and belief in me provided the strength needed to persevere through challenges.

Karlsruhe, February 2024

Yinghan Zhao

Contents

Chapter 1

Introduction

Materials science, a multidisciplinary field that explores the design, characterization, and utilization of materials, usually relies on empirical and theoretical approaches to discover novel materials for a wide range of applications in different areas [1, 2]. Currently, the development of new materials is mainly based on the researchers' scientific expertise and a large number of repetitive "trial-and-error" experiments, necessitating often a combination of professional domain knowledge and serendipity as well as luck. However, given the competitive nature of the manufacturing industry and the rapid pace of economic development, materials scientists and engineers must improve their research methodology to address the challenges of shortening the development cycles from discovery to real application of novel materials.

Experimental measurements involving the synthesis and analysis of the relationship between process, structure, and property offer a traditional approach to materials research. However, such experiments typically require specialized equipment, controlled environments, and a high level of researcher expertise, which can be time-consuming and less efficient. In contrast, computational methods at different scales, including density functional theory [3, 4], molecular dynamics [5], Monte Carlo techniques [6, 7], the phase-field method [8, 9, 10], and macroscopic continuum approaches [11], can provide an alternative way for studying materials using computer programs. Their use in computational materials design is expected to uncover new materials and reduce the time and cost of materials development [12]. Compared to experimental measurements, computational simulations can offer full control over relevant variables, positioning them as good substitutes for real experiments. Despite these advantages, computational simulations still face challenges such as a strong dependence on physical/chemical laws behind the studied system and the need for high-performance computing clusters. In modern materials research, there is a growing trend towards the integration of computational and experimental methods for a comprehensive understanding of the structural and chemical/physical properties of materials. However, limitations in both experimental methods and theoretical calculations hinder the efficiency in accelerating materials discovery and design.

Since the 21st century, the advancements in information technology have ushered in the era of “big data” [13]. The study of materials has always been an active field with a long history, and research activities have resulted in a colossal accumulation of valuable data. However, the collection and structured organization of generated data have been inadequate due to the broad and heterogeneous variety of research, which prevents their efficient use. The scientific community is now actively engaged in addressing this issue and considerable efforts have been made by materials scientists to compile comprehensive data sets of material properties such as Inorganic Crystal Structure Database (ICSD) [14] and Materials Project (MP) [15]. These data sets have the potential to accelerate the materials discovery and design process [16], and have been proven effective for a variety of material systems such as energy materials [17], porous materials [18], polymer materials [19]. In addition, advances in characterization techniques are generating large amounts of data, rendering traditional evaluation methods ineffective, and creating challenges for scientists to analyze the information promptly. As a result, researchers need new methods to analyze and evaluate the data generated in their studies [20].

Machine Learning (ML) [21] is becoming a transforming force in the ever-evolving field of materials science, offering a vast and exciting landscape of innovation and discovery. The main goal of ML is to design and analyze learning algorithms that allow computers to automatically analyze and extract patterns from data. These learned patterns can be used to predict unknown data, assisting humans in decision-making and performing specific tasks. Therefore, they are promising in aiding researchers to study materials and accelerate the development of novel materials. The study of ML involves a wide range of disciplines, such as statistics, mathematical optimization, and computational science. The current trend in ML is largely attributed to the rapid emergence of neural network models [22], which have demonstrated remarkable performance in applications including speech recognition [23] as well as image classification [24], and are showing great potentials in solving practical problems in materials research [1, 25, 26].

The development of materials usually involves preparing specimens through experimentation, identifying their structure, and measuring their properties using various characterization and testing techniques. Additionally, computational simulations are often performed to investigate the underlying mechanisms. ML can be utilized in these three fundamental components of materials research. The advanced ability of ML to analyze large data sets and identify complex patterns offers a promising solution for addressing various challenges of materials development. ML can benefit many aspects in the field of materials science and many studies have proven the ability of ML to significantly accelerate the process of developing materials [27, 28, 29]. Taking advantage of its ability to build accurate predictive models by learning complex relationships among data in an automatic manner, ML can be utilized to capture the relationship between materials and their corresponding properties. Once such a function of mapping from material to its property is built, it can be used to predict various material properties with satisfactory accuracy, which allows for screening good candidate materials over a large search space [30]. Since the prediction process is very fast, ML can also be used to accelerate computational simulations by predicting results that are expensive and time-consuming to calculate [31]. In addition, ML's capability to identify patterns among large data sets makes it appropriate for analyzing and interpreting complex, high-dimensional characterization data [32]. Apart from this, ML is also commonly employed to support decision-making, therefore it can realize the design of experiments and determination of optimal parameters for simulations, aiming to reduce the required number of trials and streamline standard workflows for optimization [33]. A close collaboration between ML and materials science has the potential to revolutionize the way materials are discovered, designed, and optimized, ultimately leading to breakthroughs with significant implications for numerous industries and technologies.

This work proposes a unified workflow for applying machine learning in materials research and explores its multifaceted applications through three compelling use cases. Each use case demonstrates the significant impact and effectiveness of this methodology in the process of material development, including the design of experiments, prediction of simulations, and enhanced analysis of characterization data. As a starting point, an introduction to ML is provided in Chapter 2, followed by the proposal of the workflow for applying machine learning in materials science which is illustrated by a comprehensive overview of different ML applications examples in materials research in Chapter 3. These examples explore a variety of topics such as the discovery of novel materials, prediction of their properties, design of experiments, optimization as well as acceleration of simulations, and assistance in the analysis of heterogeneous characterization data. In the next step, three prototypical studies are presented in Chapter 4 to demonstrate the advantages of using ML to facilitate materials research, encompassing the fundamental research elements that are essential for material studies: experiments, characterization, and simulation. Finally, the learned experience and gained reflections during these studies are summarized and discussed in Chapter 5.

Chapter 2

Theoretical foundations of machine learning

In this chapter, the basic concepts, essential elements of Machine Learning (ML), and most used models will be introduced. Although there are various kinds of ML algorithms, they usually consist of three basic elements: model, learning strategy (criterion), and optimization algorithm [21]. Most ML algorithms can be considered as different combinations of these three basic elements. The most dominant classes of methods in ML nowadays are statistical learning methods which treat ML problems as statistical inference problems, and the so-called Deep Learning (DL) models which are primarily based on (artificial) neural networks [34].

2.1 Basic concepts

Machine Learning (ML) is an automatic way of learning knowledge (or laws) by computers from data. As a discipline, ML typically refers to a set of methods used to solve problems, such as identifying patterns in observed data (samples), and utilizing the learned patterns (models) to make predictions on unknown or unforeseen data. A widely accepted standard definition of ML can be described as follows: a program is considered to have learned from experience E in a given task T if its performance (measured by P) at tasks in T can improve with experience E, then it is named as machine learning [21]. According to this view, ML is the method by which a computer system improves its performance through the use of data and statistical methods.

In the early days of engineering, ML was also often referred to as Pattern Recognition (PR) [35], which was more oriented to specific application tasks such as optical character recognition, speech recognition, and face recognition. These tasks are typically easy for humans to perform, but it can be challenging to design a computer program to perform them since humans may not always understand how they accomplish these tasks. Take handwritten digit recognition as an example, that is, making the computer automatically recognize the handwritten digits. Handwritten number recognition is a classical ML task that is simple for humans but very challenging for computers. It is difficult to summarize the handwriting characteristics of each number and to design the rule to distinguish different numbers, so designing a recognition algorithm is an extremely difficult task. In real life, many problems are similar to this, such as object recognition and speech recognition. For such problems, it is usually difficult to design a computer program to solve them, and even if some heuristic rules can be implemented, the process is extremely complicated. Therefore, ML appears as an alternative solution with the idea of designing an algorithm that allows the computer to learn some rules from samples and to perform tasks in an automatic manner. For example, a large number of handwritten digit images can be manually labeled to train a learning algorithm that generates a reliable model for recognizing new handwritten digits. This machine learning process shares similarities with human learning, which is the same process used to teach children to recognize numbers.

To better describe the concepts of ML and the problems it solves, some basic notations and definitions are given below. The key principle of ML is to learn knowledge through data, and a collection of samples (instances) is referred to as a data set. Generally, the data set can be divided into different parts. Some parts are used to train the ML model to learn the rules from the data (training set), while the remaining parts are used for adjusting the model's settings (validation set) or testing the model's performance (testing set). Usually, a D-dimensional vector (called feature vector) $\boldsymbol{x} = [x_1, x_2, \cdots, x_D]^\intercal$ is used to represent features of a sample, where each dimension represents one feature. The label of that sample is usually represented by the scalar value y.

Assume that the training set $\mathcal{D}$ consists of N samples, each of which is Independently and Identically Distributed (IID), i.e., drawn independently from the same data distribution, denoted $\mathcal{D} = \{(\boldsymbol{x}^{(1)}, y^{(1)}), (\boldsymbol{x}^{(2)}, y^{(2)}), \cdots, (\boldsymbol{x}^{(N)}, y^{(N)})\}$. Given a training set $\mathcal{D}$, the computer needs to automatically find an "optimal" function (model) $f^*(\boldsymbol{x})$ from a set of functions $\mathcal{F} = \{f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), \cdots\}$ to approximate the true mapping between the feature vector $\boldsymbol{x}$ and the label y for each sample. For a sample $\boldsymbol{x}$, the value of its label y can be predicted by the function $\hat{y} = f^*(\boldsymbol{x})$, or through the conditional probability of the label $\hat{p}(y \,|\, \boldsymbol{x}) = f_y^*(\boldsymbol{x})$. How to find this "optimal" function $f^*(\boldsymbol{x})$ is the key to machine learning, which is usually done by some optimization algorithms and this search process is often called the training process. To evaluate the quality of the model, a set of samples is needed to be taken independently and identically from the same distribution of the training set as the test set $\mathcal{D}'$. This set is used to evaluate the performance of the model, which serves as a criterion for guiding the process of learning a good-performing model.

2.2 Essential elements

Machine learning methods can be summarized as follows: (i) starting from a given, limited set of training data, assuming that the data are independently and identically distributed and assuming that the model to be learned belongs to a certain function set called hypothesis space; (ii) applying a certain criterion to evaluate the performance of the model; (iii) learning an optimal model from the hypothesis space (which is achieved by some optimization algorithms), so that it has the best performance on both the known training data and unknown test data [36]. In this way, the learning process consists of a hypothesis space for the model, a criterion for model selection, and an algorithm for model learning. These are called the three elements of a machine learning method: model, learning criteria, and optimization algorithm.

2.2.1 Model

For a machine learning task, the first step is to determine its input space $\mathcal{X}$ and output space $\mathcal{Y}$. The type of output space is the main difference between machine learning tasks. In the binary classification problem, $\mathcal{Y} = \{+1, -1\}$, in the multiclassification problem with C different classes, $\mathcal{Y} = \{1, 2, \cdots, C\}$, and in the regression problem the output space is usually the real number space $\mathbb{R}$, that is, $\mathcal{Y} = \mathbb{R}$. The input space $\mathcal{X}$ and the output space $\mathcal{Y}$ form a sample space. For a sample $(\boldsymbol{x}, y) \in \mathcal{X} \times \mathcal{Y}$ in the sample space, it is assumed that the relationship between $\boldsymbol{x}$ and y can be described by an unknown true mapping function $y = g(\boldsymbol{x})$ or a true conditional probability distribution $p_r(y \mid \boldsymbol{x})$. The goal of machine learning is to find a model that approximates this true mapping function $g(\boldsymbol{x})$ or the true conditional probability distribution $p_r(y \mid \boldsymbol{x})$. Since the exact form of the true mapping function $g(\boldsymbol{x})$ or the conditional probability distribution $p_r(y \mid \boldsymbol{x})$ is unknown, a set of functions $\mathcal{F}$ needs to be assumed empirically, called hypothesis space. Next, an ideal hypothesis $f^* \in \mathcal{F}$ from it is selected by observing its properties on the training set $\mathcal{D}$. Assume that the space $\mathcal{F}$ is usually a parameterized family of functions $\mathcal{F} = \{f(\boldsymbol{x}; \boldsymbol{\theta}) \mid \boldsymbol{\theta} \in \mathbb{R}^D\}$, where $f(\boldsymbol{x}; \boldsymbol{\theta})$ is a model (function) with parameters $\boldsymbol{\theta}$, and D is the number of parameters. Hypothesis spaces can generally be divided into linear and nonlinear ones, and the corresponding models f are called linear and nonlinear models, respectively.

Linear models The hypothesis space of the linear model is a parameterized family of linear functions, i.e., $f(\boldsymbol{x}; \boldsymbol{\theta}) = \boldsymbol{w}^\intercal \boldsymbol{x} + b$, where the parameters $\boldsymbol{\theta}$ contain the weight vector $\boldsymbol{w}$ and the bias b.

Nonlinear models The generalized nonlinear model can be written as a linear combination of multiple nonlinear basis functions $\phi(\boldsymbol{x})$: $f(\boldsymbol{x};\boldsymbol{\theta}) = \boldsymbol{w}^\intercal\phi(\boldsymbol{x}) + b$, where $\phi(\boldsymbol{x}) = [\phi_1(\boldsymbol{x}), \phi_2(\boldsymbol{x}), \cdots, \phi_K(\boldsymbol{x})]^\intercal$ is a vector of K nonlinear basis functions, and the parameters $\boldsymbol{\theta}$ contain the weight vector $\boldsymbol{w}$ and the bias b. If $\phi(\boldsymbol{x})$ is itself a learnable basis function, for example, $\phi_k(\boldsymbol{x}) = h(\boldsymbol{w}_k^\intercal\phi'(\boldsymbol{x})+b_k), \forall 1 \leq k \leq K$, where $h(\cdot)$ is a nonlinear function, $\phi'(\boldsymbol{x})$ is another set of basis functions, and $\boldsymbol{w}_k$ and b_k are learnable parameters, then $f(\boldsymbol{x};\boldsymbol{\theta})$ is equivalent to a neural network model [34].

2.2.2 Learning criteria

Let the training set $\mathcal{D} = \{(\boldsymbol{x}^{(n)}, y^{(n)})\}_{n=1}^N$ be composed of N IID samples, i.e., each sample $(\boldsymbol{x}, y) \in \mathcal{X} \times \mathcal{Y}$ is generated independently and randomly from the joint space of $\mathcal{X}$ and $\mathcal{Y}$ according to some unknown distribution $p_r(\boldsymbol{x}, y)$. Here the sample distribution $p_r(\boldsymbol{x}, y)$ must be fixed. The learning criteria is that, a good model $f(\boldsymbol{x}, \boldsymbol{\theta}^*)$ should be consistent with the true mapping function $y = g(\boldsymbol{x})$ for all possible values of $(\boldsymbol{x}, y)$, which can be measured by an evaluating metric, $\big|f(\boldsymbol{x}, \boldsymbol{\theta}^*) - y\big| < \varepsilon$, $\forall(\boldsymbol{x}, y) \in \mathcal{X} \times \mathcal{Y}$, or is consistent with the true conditional probability distribution $p_r(y|\boldsymbol{x})$, i.e., $|f_y(\boldsymbol{x}, \boldsymbol{\theta}^*) - p_r(y \,|\, \boldsymbol{x})| < \varepsilon$, $\forall(\boldsymbol{x}, y) \in \mathcal{X} \times \mathcal{Y}$, where ε is a small positive number and $f_y(\boldsymbol{x}, \boldsymbol{\theta}^*)$ is the probability corresponding to y in the conditional probability distribution predicted by the model. It should be noted that KL-divergence [37] or cross-entropy [38] is a better metric for measuring the similarity of the two distributions.

The performance of the model $f(\boldsymbol{x};\boldsymbol{\theta})$ can be measured by the expected risk $\mathcal{R}(\boldsymbol{\theta})$, which is defined as $\mathcal{R}(\boldsymbol{\theta}) = \mathbb{E}_{(\boldsymbol{x},y)\sim p_r(\boldsymbol{x},y)}\Big[\mathcal{L}\big(y, f(\boldsymbol{x};\boldsymbol{\theta})\big)\Big]$, where $p_r(\boldsymbol{x}, y)$ is the true (real) data distribution, $\mathcal{L}(y, f\big(\boldsymbol{x};\boldsymbol{\theta})\big)$ is the loss function, which is used to quantify the difference between the two variables. A loss function is a non-negative real function that quantifies the difference between the model prediction and the true label. Some commonly used loss functions are listed below:

- 0-1 loss function: $\mathcal{L}\big(y, f(\boldsymbol{x})\big) = \begin{cases} 1, y \neq f(\boldsymbol{x}) \\ 0, y = f(\boldsymbol{x}) \end{cases}$
- Quadratic loss function: $\mathcal{L}\big(y, f(\boldsymbol{x})\big) = \big(y - f(\boldsymbol{x})\big)^2$
- Absolute loss function: $\mathcal{L}\big(y, f(\boldsymbol{x})\big) = |y - f(\boldsymbol{x})|$
- Logarithmic (or log-likelihood) loss function: $\mathcal{L}\big(y, p(y \,|\, \boldsymbol{x})\big) = -\log p(y \,|\, \boldsymbol{x})$
- Hinge loss function: $\mathcal{L}\big(y, f(\boldsymbol{x})\big) = \max\big(0, 1 - yf(\boldsymbol{x})\big)$

Risk minimization criteria

A good model $f(\boldsymbol{x};\boldsymbol{\theta})$ should have a relatively small expected error, however, since the true data distribution and mapping function are unknown, it is practically impossible to calculate its (ground-truth) expected risk $\mathcal{R}(\boldsymbol{\theta})$. However, given a training set $\mathcal{D} = \{(\boldsymbol{x}^{(n)}, y^{(n)})\}_{n=1}^{N}$, the empirical risk can be calculated, which is the average loss on the training set:

$$\mathcal{R}_{\mathcal{D}}^{emp}(\boldsymbol{\theta}) = \frac{1}{N}\sum_{n=1}^{N}\mathcal{L}(y^{(n)}, f(\boldsymbol{x}^{(n)};\boldsymbol{\theta})). \tag{2.1}$$

Therefore, a practical learning criterion is to find a set of parameters $\boldsymbol{\theta}^*$ that minimizes the empirical risk, i.e., $\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \mathcal{R}_{\mathcal{D}}^{emp}(\boldsymbol{\theta})$, which is the Empirical Risk Minimization (ERM) criterion.

According to the law of large numbers, when the size of the training set $|\mathcal{D}|$ approaches infinity, the empirical risk converges to the expected risk (or expected loss). When the model is a conditional probability distribution and the loss function is a logarithmic loss function, empirical risk minimization is equivalent to Maximum Likelihood Estimation (MLE) [22]. However, typically, access to an infinite number of training samples is not available, and the training samples often constitute only a small subset of the real data or contain some noisy data, which do not represent the true distribution of the entire data set. The principle of ERM can result in a low error on the training set, but a large error on the unknown data, which is known as overfitting, that is, poor generalization ability of the model. The issue of overfitting can often arise due to limited training data, noise, and overly powerful models. To address this, a common practice is to introduce parameter regularization based on ERM, which limits the model's capability to prevent over-minimizing empirical risk. This approach is known as the Structure Risk Minimization (SRM) criterion:

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \mathcal{R}_{\mathcal{D}}^{struct}(\boldsymbol{\theta}) = \arg\min_{\boldsymbol{\theta}} \mathcal{R}_{\mathcal{D}}^{emp}(\boldsymbol{\theta}) + \frac{1}{2}\lambda\|\boldsymbol{\theta}\|^2 \tag{2.2}$$

where $\|\boldsymbol{\theta}\|$ is the L_2-norm, serving as the regularization term to reduce the parameter space and mitigate overfitting, and λ is the strength of the regularization. Another commonly used regularization term is the L_1-norm regularization, which usually results in the sparsity of the parameters and is useful in some specific situations. In Bayesian learning, regularization refers to introducing a prior distribution of parameters that is not entirely reliant on the training data, for example, Maximum A Posteriori (MAP) estimation is an illustration of the SRM principle when the model is a conditional probability distribution, the loss function is a logarithmic loss function, and the model complexity is represented by the prior probability of the model [22].

The opposing concept to overfitting is underfitting, where the model does not adequately fit the training data and has large errors in the training set. Underfitting is typically caused by insufficient modeling capability. Examples of underfitting and overfitting are given in Figure 2.1.

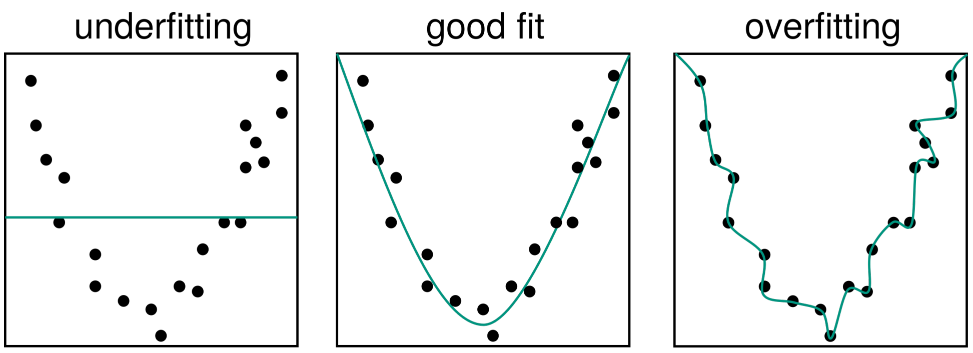


Figure 2.1: Example cases for underfitting, good fit, and overfitting, where black points are samples and fitted function is marked in the green line.

Model selection and validation

A good model should perform well on both the training data set and the test data set, that is, ML aims to not only fit the data in the training set but also to minimize error on unknown test samples (generalization ability). This is achieved by finding an optimal model with low generalization error from the hypothesis space, using the data at hand. The ultimate goal is to make accurate predictions for unforeseen samples that are not part of the training set. Thus, machine learning may be considered as the challenge of generalizing from limited, high-dimensional, and noisy data to more comprehensive patterns.

A good model should perform well on both the training data set and the test data set. When faced with a hypothesis space containing models of varying complexity, the task of selecting an appropriate model arises. It is desired to select or learn an appropriate model that can approximate the "true" model as much as possible. Figure 2.2 depicts the relationship between the training error and testing error and the complexity of the model. As the complexity of the model increases, its fitting ability also improves and the training error decreases continuously, while the testing error initially decreases, reaches a minimum value, and then increases (overfitting). The occurrence of overfitting happens when the model's complexity is too high, and therefore, overfitting prevention is necessary during the learning process.

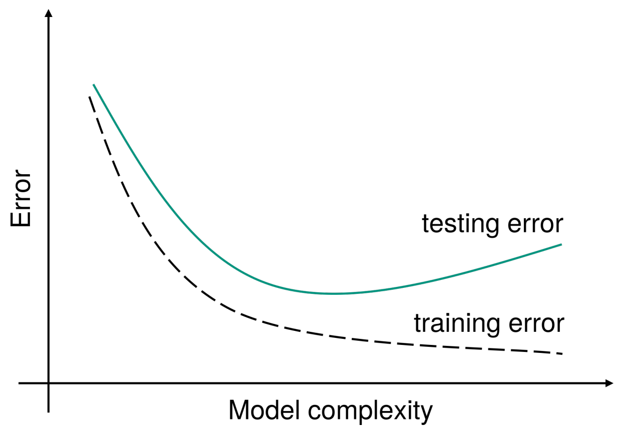


Figure 2.2: Training error and testing error against model complexity.

If the sample data is adequate, one straightforward approach for evaluating the generalization ability of the ML model is to randomly divide the data set into three parts: the training set, validation set, and test set, followed by training the model on the training set, tuning the settings of the model for selecting the optimal one, and assessing the performance of the learning method with the test set. In many practical situations, insufficient data could hinder the above validation process, prompting the use of cross-validation techniques. Cross-validation involves repetitively utilizing the data by dividing it into sliced sets, which are then merged to form a training set and a testing set, and are iteratively used for training, testing, and model selection. The most widely used cross-validation method is the K-fold cross-validation, which is as follows. First, randomly divide the provided data into K separate and equal subsets. Then, train the model using the data in $K-1$ subsets and test it using the remaining subset. Repeat this step for all possible choices of K. Finally, select the model with the lowest error in the test set among the K choices. This process is repeated for K possible selections, and the model with the smallest average test error among the K evaluations is chosen as the best-performing one.

2.2.3 Optimization algorithm

After determining the training set $\mathcal{D}$, the hypothesis space $\mathcal{F}$ and the learning criterion, the process of obtaining the optimal model $f(\boldsymbol{x}, \boldsymbol{\theta}^*)$ can be formulated as an optimization problem. The training process of ML is the process of solving the optimization problem. In ML, optimization can be divided into parametric optimization and hyperparametric optimization. The parameters $\boldsymbol{\theta}$ in the model $f(\boldsymbol{x}, \boldsymbol{\theta})$ are learned by the optimization algorithm. In addition to the learnable parameters $\boldsymbol{\theta}$, there are other parameters that are used to define the model structure or optimization strategy, called hyperparameters.

Commonly seen hyperparameters include the number of categories in clustering algorithms, the regularization term coefficients, the number of layers in a neural network, the step size in gradient descent, and the kernel function in a support vector machine, etc. The process of selecting hyperparameters typically presents a combinatorial optimization challenge that optimization algorithms struggle to learn automatically. Accordingly, optimization of hyperparameters represents a largely empirical ML technique, typically established through human experience or by searching for sets of hyperparameter combinations through trial and error.

In order to take advantage of some efficient and mature optimization methods in convex optimization, many ML methods aim to construct a convex function as the optimization objective by selecting an appropriate model and loss function. However, some models, such as neural networks, have non-convex optimization objectives which usually offer sub-optimal solutions. In ML, the gradient descent method serves as one of the most basic and prevalent optimization algorithms [34], which first initializes the parameters $\boldsymbol{\theta}_0$ and then computes the minimum value of the risk function on the training set $\mathcal{D}$ according to the following iterative equation:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \alpha \frac{\partial \mathcal{R}_{\mathcal{D}}(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = \boldsymbol{\theta}_t - \alpha \frac{1}{N} \sum_{n=1}^{N} \frac{\partial \mathcal{L}\big(y^{(n)}, f(\boldsymbol{x}^{(n)}; \boldsymbol{\theta})\big)}{\partial \boldsymbol{\theta}} \tag{2.3}$$

where $\boldsymbol{\theta}_t$ are the parameter values at the t-th iteration and α is the search step size, which is also generally referred to as the learning rate.

Stochastic gradient descent and mini-batch gradient descent method

In Equation 2.3, the objective function corresponds to the risk function of the entire training set. This specific method is referred to as Batch Gradient Descent (BGD). During each iteration, BGD calculates the gradient of the loss function for each sample and sums them over the whole data set. However, when the training set has a large number of samples, both the space complexity and computational overhead per iteration increase significantly. In order to decrease computational complexity during each iteration, it is also possible to take only one sample at each iteration, compute the gradient of the loss function of this sample, and update the parameters. This method is referred to as Stochastic Gradient Descent (SGD), which can effectively converge to a local optimum after an adequate number of iterations [39]. The difference between BGD and SGD lies in their optimization objectives: BGD calculates the average loss function across all samples, whereas SGD focuses on the loss function for a single sample. SGD is a popular choice due to its ease of implementation and rapid convergence, and it can be equivalently interpreted as introducing random noise into the gradient of the BGD method. Moreover, SGD is advantageous in non-convex optimization challenges as it is less prone to getting trapped in local optimums.

However, a drawback of SGD is its limited ability to take advantage of parallel computing. Mini-Batch Gradient Descent (mini-BGD) proves to be a viable alternative, striking a balance between BGD and SGD. In every iteration, a few training samples are randomly chosen (typically set to the power of 2 for greater computational efficiency) to compute the gradient and update the parameters, enabling the utilization of the random gradient descent technique and enhancing training efficacy. In the t-th iteration, a subset $\mathcal{S}_t$ containing K samples is randomly selected. The gradient of the loss function is calculated and averaged for each sample in this subset, followed by updating the parameters:

$$\boldsymbol{\theta}_{t+1} \leftarrow \boldsymbol{\theta}_t - \alpha \frac{1}{K} \sum_{(\boldsymbol{x},y)\in\mathcal{S}_t} \frac{\partial \mathcal{L}(y, f(\boldsymbol{x};\boldsymbol{\theta}))}{\partial \boldsymbol{\theta}}. \tag{2.4}$$

In practice, small-batch stochastic gradient descent offers fast convergence and low computational expense, making it the primary optimization algorithm for large-scale machine learning [40].

2.3 Feature representation

In practical applications, there are various types of data, including text, audio, image, and video. The space of raw features varies for different types of data. For instance, a grayscale image (with a pixel count of D) has a feature space defined by $[0, 255]^D$. To ensure that input sample features are mathematically computable, numerous machine learning algorithms necessitate converting these different data types into vector representations before fitting ML models. Using the original data features to make predictions necessitates a greater modeling capability. However, these raw features can have certain drawbacks: (i) they may be monotonous and require non-linear combination for better usability; (ii) there can be high redundancy amongst features; (iii) not all features may be applicable for prediction; (iv) features may contain noise. The benefits of feature selection and extraction lie in their ability to convey the most relevant information from the original features utilizing fewer parameters, filter out unnecessary noise, and optimize computational efficiency while decreasing dimensionality challenges. Consequently, feature selection and extraction remain vital for numerous non-regularized models. After selecting or extracting features, the number of features is typically reduced. This process is often referred to as dimensionality reduction.

2.3.1 Traditional learning of features

Traditional feature learning is commonly achieved by designing specific criteria and subsequently choosing effective features based on those criteria. This process is generally categorized into two types: feature selection and feature extraction. The advantage of feature selection and feature extraction is that they can represent most of the relevant information in the original features with fewer features, remove the noise information, and thus improve the computational efficiency and reduce the curse of dimensionality [41]. For many models without regularization, feature selection and feature extraction are necessary.

Feature selection

Feature selection is the procedure for choosing a good subset from the original feature set to yield the best-performing model. In essence, feature selection entails retaining only helpful features while discarding the redundant or insignificant ones. One approach to finding the most accurate subset of features in a machine learning model is to test each possible subset. However, this method can be incredibly complex and inefficient, as there can be up to 2^D candidate subsets supposing the original feature set contains D elements. A more commonly used strategy involves a greedy approach, which starts with an empty set and adds the best features in each round, known as "forward search", or begins with the original set of features and removes the most useless features in each round, referred to as "backward search". Subset search methods can be categorized into filtering methods and wrapping methods.

- The filtering method is a feature selection technique that does not depend on any particular machine learning model. It operates by either adding the most informative features or removing the least informative ones at each iterative stage [42]. The information content of a feature can be quantified using information gain, which signifies the reduction in the uncertainty or entropy of the $\boldsymbol{\theta}$-parameterized conditional distribution $p_{\boldsymbol{\theta}}(y \mid \boldsymbol{x})$ upon the incorporation of the feature.
- The wrapping method evaluates subsequent machine learning models by their accuracy [36]. It adds the features that are most helpful for the next machine learning model or removes the ones that are least helpful for the subsequent task. This approach encompasses the integration of the machine learning model into the process of feature selection.

Besides, the L_1 regularization can also be used to select features since L_1 regularization leads to sparse features and thus indirectly achieves feature selection [36].

Feature extraction

Feature extraction is the construction of a new feature space by projecting the original features into the new space to obtain a new representation. Feature extraction can be separated into two categories: supervised and unsupervised methods. The objective of supervised feature learning is to extract the most advantageous features for a specific prediction task, for example, Linear Discriminant Analysis (LDA) [43]. Meanwhile, the goal of unsupervised feature learning is usually to diminish redundant information and noise, using techniques like Principal Component Analysis (PCA) [44, 45] and AutoEncoders (AEs) [46].

2.3.2 Deep learning of features

Traditional feature learning is generally separated from the learning process of the prediction model. For example, effective features can be first extracted through PCA or LDA and then a specific machine learning model can be trained based on these effective features. If the process of learning feature representations and the training of machine learning models are integrated into one framework with an end-to-end learning algorithm, it has the potential to eliminate inconsistencies between their respective criteria. This method of representation learning is known as Deep Learning (DL) [22]. The challenge of the DL approach lies in evaluating the impact of representation learning on the final system output, which is commonly referred to as the contribution allocation problem. Neural network models are currently the most efficient models as they utilize the final output layer for prediction learning and employ the other layers for representation learning [34].

2.4 Evaluation metrics

To assess the performance of a machine learning model, an evaluation score is calculated based on the prediction results by applying the model to a test set. The method with a lower test error is viewed as being more powerful and has better predictive ability.

For regression tasks, one of the most commonly used evaluation metrics is the Mean Squared Error (MSE), which measures the average of the squares of the errors, that is, the average squared difference between the estimated values from the model and the true values (labels). Given a data set $\mathcal{D} = \{(\boldsymbol{x}^{(1)}, y^{(1)}), (\boldsymbol{x}^{(2)}, y^{(2)}), \cdots, (\boldsymbol{x}^{(N)}, y^{(N)})\}$, assume that the label $y \in \mathbb{R}$, and the learned model $f(\boldsymbol{x}; \boldsymbol{\theta}^*)$ is used to predict each sample in the data set with $\{\hat{y}^{(1)}, \cdots, \hat{y}^{(N)}\}$. The MSE can be calculated with:

$$\mathrm{MSE} = \frac{1}{N} \sum_{n=1}^{N} (y^{(n)} - \hat{y}^{(n)})^2. \tag{2.5}$$

For classification tasks, widely used evaluation metrics are accuracy, precision, recall, and F-measure, which can be defined as follows (assuming the labels within C classes $y \in \{1, \cdots, C\}$).

Accuracy: the most commonly used evaluation metric is accuracy:

$$Acc = \frac{1}{N}\sum_{n=1}^{N} I(y^{(n)} = \hat{y}^{(n)}) \tag{2.6}$$

where $I(\cdot)$ is the indicator function whose value is 1 if true or 0 if false.

Error rate: the error rate corresponds to the accuracy:

$$Err = 1 - Acc = \frac{1}{N}\sum_{n=1}^{N} I(y^{(n)} \neq \hat{y}^{(n)}). \tag{2.7}$$

Precision and recall: accuracy represents the mean performance of all classes, and to estimate the performance of each class, it is necessary to calculate both precision and recall, which are prominent metrics in machine learning evaluation. When examining category c, the model's results on the data set can be categorized into one of four categories:

1. True Positive (TP): a sample belongs to true class c and the model correctly predicts it to be c. The number of samples in this category is denoted as

$$TP_c = \sum_{n=1}^{N} I(y^{(n)} = \hat{y}^{(n)} = c). \tag{2.8}$$

2. False Negative (FN): a sample with the true class of c is incorrectly classified by the model as belonging to a different class. The number of samples in this category is denoted as

$$FN_c = \sum_{n=1}^{N} I(y^{(n)} = c \wedge \hat{y}^{(n)} \neq c) \tag{2.9}$$

where $\wedge$ is the logical conjunction.

3. False Positive (FP): a sample is mistakenly classified as class c while its true class is not. The number of samples in this category is denoted as

$$FP_c = \sum_{n=1}^{N} I(y^{(n)} \neq c \wedge \hat{y}^{(n)} = c). \tag{2.10}$$

4. True Negative (TN): a sample belongs to another class, and the model predicts it to be others. The number of samples in this category is denoted as TN_c. For category c, this case is generally of few concern. In practical studies, it is often possible to form the confusion matrix of the above four to facilitate the comparison of relationships.

Based on the above definition, the precision, recall, and F-measure can be further defined:

- Precision: the percentage of correct predictions for category c among all samples predicted for category c
$$\mathcal{P}_c = \frac{TP_c}{TP_c + FP_c}. \tag{2.11}$$
- Recall: the percentage of correct predictions for category c among all samples whose true label is category c
$$\mathcal{R}_c = \frac{TP_c}{TP_c + FN_c}. \tag{2.12}$$
- F-measure: a composite metric that is a summed average of the precision and recall
$$\mathcal{F}_c = \frac{(1+\beta)^2 \times \mathcal{P}_c \times \mathcal{R}_c}{\beta^2 \times \mathcal{P}_c + \mathcal{R}_c} \tag{2.13}$$
where β is used to balance the importance of precision and recall and is usually taken to be 1. The F-measure is also called the F_1-value when $\beta = 1$ and is the summed average of precision and recall.

In practice, the thresholds of the classification models can be adjusted for a more comprehensive evaluation, such as Area Under Curve (AUC), Receiver Operating Characteristic (ROC) curve, and Precision-Recall (PR) curve [36]. In addition, many tasks have their own specialized evaluation methods, such as top-N accuracy.

2.5 Types of machine learning models

Machine learning algorithms can be classified according to different criteria. However, in general, they are usually classified into the following categories according to the information provided by the training samples and the feedback methods [36].

2.5.1 Supervised learning

If the goal of machine learning is to model the relationship between the features of a sample $\boldsymbol{x}$ and its corresponding (usually a scalar value) label $y = f(\boldsymbol{x};\boldsymbol{\theta})$ or $p(y \mid \boldsymbol{x};\boldsymbol{\theta})$, and if every sample in the training set is labeled, then this type of machine learning is referred to as supervised learning. Given a training data set $\mathcal{D} = \{(\boldsymbol{x}^{(1)}, y^{(1)}), (\boldsymbol{x}^{(2)}, y^{(2)}), \cdots, (\boldsymbol{x}^{(N)}, y^{(N)})\}$, where $(\boldsymbol{x}^{(n)}, y^{(n)})$, $n = 1, 2, \cdots, N$, are called samples. The $\boldsymbol{x} \in \mathcal{X} \subseteq \mathbb{R}^D$ are the input observations, also called inputs or instances, and the $y \in \mathcal{Y}$ are the output observations, also called outputs.

Supervised learning can be divided into two processes, learning and prediction, which are performed by the learning system and the prediction system, as shown in Figure 2.3. In the learning phase, the learning system utilizes a given training data set to learn (or train) a model to map the relationship between input and output, which can be described by either the conditional probability distribution $p(y \mid \boldsymbol{x})$ or the decision function $y = f(\boldsymbol{x})$. To predict the corresponding output $y^{(N+1)}$ for a given input $\boldsymbol{x}^{(N+1)}$ in the test sample set, the prediction system utilizes the model $y^{(N+1)} = \arg\max_y p(y \mid \boldsymbol{x}^{(N+1)})$ or $y^{(N+1)} = f(\boldsymbol{x}^{(N+1)})$.

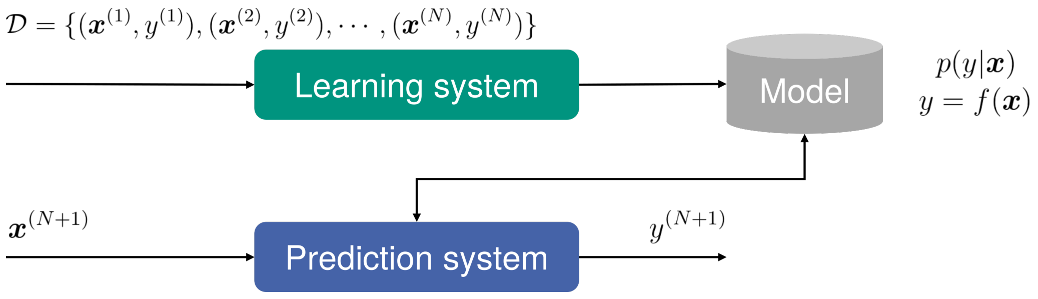

$\mathcal{D} = \{(\boldsymbol{x}^{(1)}, y^{(1)}), (\boldsymbol{x}^{(2)}, y^{(2)}), \cdots, (\boldsymbol{x}^{(N)}, y^{(N)})\}$
Learning system
Model
$p(y|\boldsymbol{x})$
$y = f(\boldsymbol{x})$
$\boldsymbol{x}^{(N+1)}$
Prediction system
$y^{(N+1)}$

Figure 2.3: Schematic illustration of supervised learning.

Depending on the type of labels, supervised learning can be divided into regression problems, classification problems, and structured learning problems:

- Regression problem has a continuous value (real or continuous integer) of label y. The output of $f(\boldsymbol{x}; \boldsymbol{\theta})$ is also a continuous value. It should be noted that regression tasks can be turned into classification tasks by setting appropriate thresholds [47, 48].

- Classification problem has a discrete category (symbol) of label y. In the classification problem, the learned model is also called a classifier. The classification problem can be divided into binary classification and multi-class classification problems according to the number of classes. It should also be noted that in many situations, classification and regression problems can be converted to each other [35], e.g., the output of a classification model can be a probability value for a class.

- Structured learning problem is a special kind of classification problem. In structured learning, the labels $\boldsymbol{y}$ are usually structured objects, such as sequences, trees, or graphs.

Some of the most popular supervised learning models are described in the following sections.

Logistic regression

Logistic Regression (LR), or logit regression [49], as one of the generalized linear models, is actually one of the classification methods in statistical learning. A schematic diagram of this model is shown in Figure 2.4.

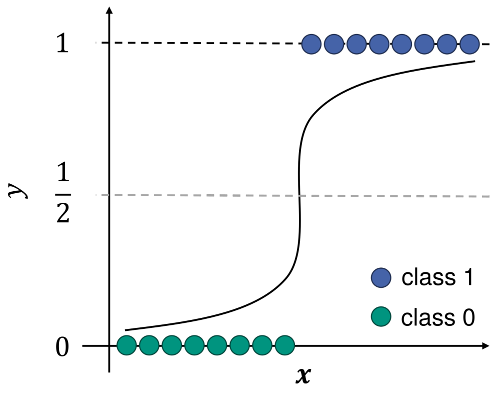


Figure 2.4: Example of a binary classification problem for 1-D data using logistic regression.

In the binary classification case (binomial logistic regression model), it is used to model the probability that a given example belongs to the class “1” versus the probability that it belongs to the class “0” (usually noted as $y \in \{0, 1\}$). For this statistical model, a linear relationship is assumed between the predictor variables and the log-odds (also called logit, odds means the ratio of probability between $y = 1$ and $y = 0$). This linear relationship can usually be written in the following mathematical form:

$$\log(\text{odds}) := \ln \frac{p}{1-p} = \ln \frac{p(y=1 \mid \boldsymbol{x})}{1 - p(y=1 \mid \boldsymbol{x})} = \boldsymbol{w}^\intercal \boldsymbol{x} + b \tag{2.14}$$

where the $\boldsymbol{x} \in \mathbb{R}^D$ are the inputs, the weights $\boldsymbol{w} \in \mathbb{R}^D$ and the bias (intercept) $b \in \mathbb{R}$ are parameters. The odds can be recovered by exponentiating the log-odds

$$\frac{p}{1-p} = e^{\boldsymbol{w}^\intercal \boldsymbol{x} + b} \tag{2.15}$$

and by some algebraic manipulations the general logistic function $p : \mathbb{R} \to (0, 1)$ can be written as:

$$p = p(y = 1 \mid \boldsymbol{x}) = \frac{e^{\boldsymbol{w}^\intercal \boldsymbol{x} + b}}{1 + e^{\boldsymbol{w}^\intercal \boldsymbol{x} + b}} \tag{2.16}$$

$$1 - p = p(y = 0 \mid \boldsymbol{x}) = \frac{1}{1 + e^{\boldsymbol{w}^\intercal \boldsymbol{x} + b}}. \tag{2.17}$$

Given training data set $\mathcal{D} = \{(\boldsymbol{x}^{(1)}, y^{(1)}), (\boldsymbol{x}^{(2)}, y^{(2)}), \cdots, (\boldsymbol{x}^{(N)}, y^{(N)})\}$, parameters $\boldsymbol{w}$ and b can be estimated with the maximum likelihood estimation. The model tries to maximize the log-likelihood

$$\mathcal{L}(\boldsymbol{w}, b) = \sum_{i=1}^{N} \ln p(y^{(i)} \mid \boldsymbol{x}^{(i)}; \boldsymbol{w}, b) \tag{2.18}$$

such that the probability of each sample belonging to its true label is as high as possible.

In the context of multi-class classification, this specific model can be extended to multinominal logistic regression. Furthermore, the One-versus-Rest (OvR) scheme can be utilized for implementing multi-category classification. The OvR scheme splits the task of multi-category classification into several binary classifications. Specifically, for each class, an LR model is trained. During the training process of each model, the selected class is identified as the positive sample while the other classes are marked as negative samples.

K-nearest neighbor

The k-Nearest Neighbor (k-NN) method is a basic classification and regression method [50]. The k-NN basically consists of determining the k-nearest neighbors of the input instance points from a set of given training instance points. The basic approach is: for a given set of training instance points and an input instance point, first determine the k nearest neighbors of the input instance point from the training instance points, and then use the majority of the classes of these k training instance points to predict the class of the input instance point (the so-called majority voting method). When employing k-NN for the regression task, the average method is typically chosen, which means the prediction is the average value of the closest k training instance points. It should be noted that the process of finding neighbors can be computationally intensive when dealing with large data sets. A schematic diagram of this model is shown in Figure 2.5.

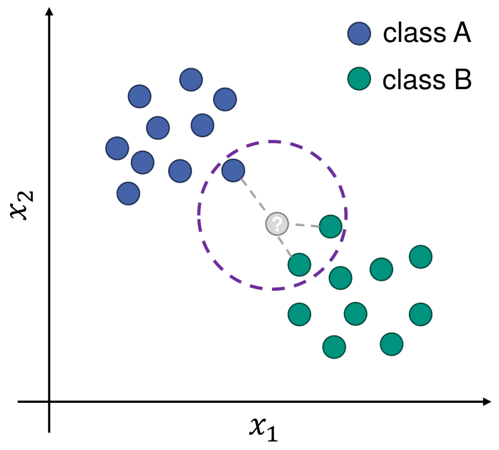


Figure 2.5: Schematic illustration of k-NN.

The k-NN model partitions the feature space using the training data set. The training set, distance metric, k-value, and classification/regression decision rule are the three basic elements of the k-nearest neighbor method, which uniquely determine the results in k-NN. Commonly used distance metrics are the Euclidean distance and the more general L_p distance [51]. The optimal k value is usually selected by cross-validation.

New instances are predicted by considering their categories/values based on their k-nearest neighbors from the training instances. Hence, the k-NN model does not have an explicit learning process. A k-NN model uses the training data set to partition the feature vector space and serves as a “model” for its prediction. Such kind of setting can make k-NN more sensitive to local data structures.

Support vector machine

Support Vector Machines (SVMs) are supervised max-margin models that can analyze data for classification and regression tasks [52]. The margin is defined as the shortest distance from all samples to the decision hyperplane of the model. Its basic form is a linear classifier defined as a margin-maximizer on the feature space, which distinguishes it from a perceptual machine [52, 53]. The original SVM was a binary classification model and was later generalized to a multi-class classification SVM [54, 55].

Given a dichotomous classification data set $\mathcal{D} = \{(\boldsymbol{x}^{(n)}, y^{(n)})\}_{n=1}^{N}$, where $y^{(n)} \in \{+1, -1\}$, if the two classes of samples are linearly separable, i.e., a hyperplane $\boldsymbol{w}^{\intercal}\boldsymbol{x} + b = 0$ can separate the two classes of samples, then for each sample $y^{(n)}(\boldsymbol{w}^{\intercal}\boldsymbol{x}^{(n)} + b) > 0$. The distance $\gamma^{(n)}$ from each sample $\boldsymbol{x}^{(n)}$ in the data set $\mathcal{D}$ to the separating hyperplane is:

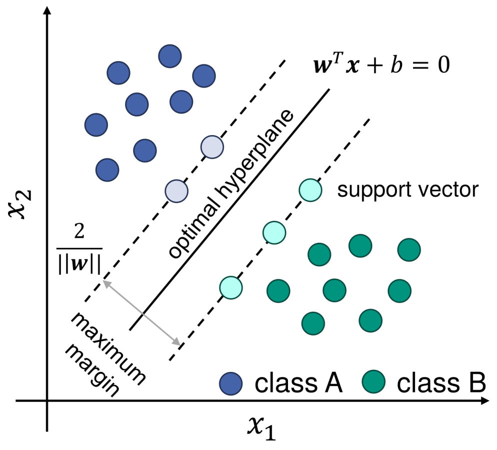
$$\gamma^{(n)} = \frac{|\boldsymbol{w}^{\intercal}\boldsymbol{x}^{(n)} + b|}{\|\boldsymbol{w}\|} = \frac{y^{(n)}(\boldsymbol{w}^{\intercal}\boldsymbol{x}^{(n)} + b)}{\|\boldsymbol{w}\|} \tag{2.19}$$

where $\|\boldsymbol{w}\|$ is the L_2-norm of $\boldsymbol{w}$.

The margin γ is defined as the shortest distance from all samples in the entire data set $\mathcal{D}$ to the separating hyperplane $\gamma = \min_{n} \gamma^{(n)}$. The larger the margin is, the more stable its separating hyperplane is for distinguishing the two classes, and less susceptible to the noise. The goal of SVM is to find a hyperplane $(\boldsymbol{w}^*, b^*)$ such that γ is maximized, i.e., $\max_{\boldsymbol{w},b} \gamma$. This can be equivalently expressed as a convex optimization form

$$\begin{aligned} \min_{\boldsymbol{w},b} \quad & \frac{1}{2}\|\boldsymbol{w}\|^2 \\ \text{s.t.} \quad & 1 - y^{(n)}(\boldsymbol{w}^{\intercal}\boldsymbol{x}^{(n)} + b) \leq 0, \quad \forall n \in \{1, \cdots, N\} \end{aligned} \tag{2.20}$$

where sample points that satisfy $y^{(n)}(\boldsymbol{w}^{\intercal}\boldsymbol{x}^{(n)} + b) = 1$ are called support vectors. A schematic diagram of this model is shown in Figure 2.6.



$\boldsymbol{w}^T\boldsymbol{x} + b = 0$
optimal hyperplane
support vector
$\frac{2}{||\boldsymbol{w}||}$
maximum margin
class A
class B
x_1
x_2

Figure 2.6: Schematic illustration of SVM.

To find the separating hyperplane with maximum margin, the primal object function Equation 2.20 can be reformulated with the Lagrange multiplier method as a Lagrangian dual function

$$\Gamma(\lambda) = -\frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N}\lambda_m\lambda_n y^{(m)}y^{(n)}\boldsymbol{x}^{(m)})^{\intercal}\boldsymbol{x}^{(n)} + \sum_{n=1}^{N}\lambda_n \tag{2.21}$$

where $\lambda_1 \geq 0, \cdots, \lambda_N \geq 0$ are Lagrange multipliers. The primal optimization problem of SVM is a convex optimization problem that satisfies strong duality, that is, the primal optimization problem can be solved by maximizing the dual function $\max\limits_{\lambda\geq 0}\Gamma(\lambda)$, which is a convex optimization problem that can be solved by a variety of convex optimization methods. The decision function of SVM with optimal parameters is

$$\begin{aligned} f(\boldsymbol{x}) &= \operatorname{sgn}(\boldsymbol{w}^{*\intercal}\boldsymbol{x} + b^*) \\ &= \operatorname{sgn}\left(\sum_{n=1}^{N}\lambda_n^* y^{(n)}(\boldsymbol{x}^{(n)})^{\intercal}\boldsymbol{x} + b^*\right) \end{aligned} \tag{2.22}$$

where sgn$(\cdot)$ is the sign function that takes the values -1,0,1 when the value of the expression in parentheses is less than, equal to, or greater than 0, respectively.

SVMs can also use kernel functions to map samples from the original feature space to a higher dimensional space. This allows them to solve linearly non-separable problems that may exist in the original feature space [56]. In addition to this, soft-margin SVM with slack variables can be introduced to solve the problem when the samples in the data set are not linearly separable in the feature space, which works by tolerating some samples that do not meet the constraints of the original objective function [52].

Gaussian process regression

Gaussian Process Regression (GPR) [57] is a Bayesian non-parametric model commonly employed for non-linear modeling. The framework uses probabilistic theory to generate estimates that have probabilistic meaning. This is accomplished by introducing randomness to the model, which can organically integrate the researcher's a priori knowledge with information learned from observational data, and reduce uncertainty through Bayesian inference. GPR models have a strong generalization ability, making them suitable for solving regression problems involving nonlinear, high-dimensional, and data with small sample sizes.

As a Bayesian non-parametric model, GPR allows for Bayesian inference in function space by using a Gaussian process as a prior. The Gaussian process is a more general version of the multidimensional Gaussian distribution, which is completely determined by the mean function $m(\boldsymbol{x})$ and covariance function $K(\boldsymbol{x}, \boldsymbol{x}')$:

$$f(\boldsymbol{x}) \sim \mathcal{GP}\big(m(\boldsymbol{x}), K(\boldsymbol{x}, \boldsymbol{x}')\big) \tag{2.23}$$

and a schematic diagram of this model is shown in Figure 2.7.

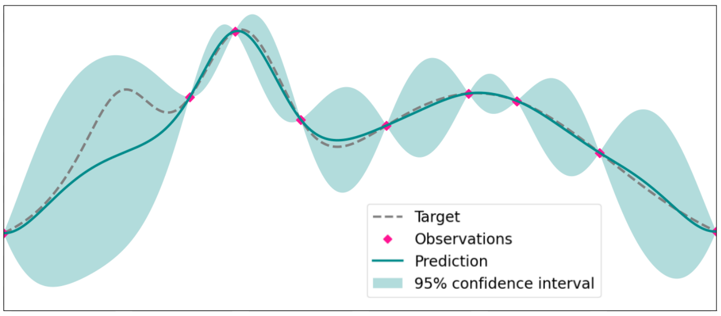


Figure 2.7: Schematic illustration of GPR, where the gray dashed lines denote the unknown function to be fitted, while the green solid line shows the mean of the GPR prediction. The shaded area represents the magnitude of uncertainty (assuming that the observed data are noise-free to facilitate the illustration), and the observed sample points are marked in pink.

Consider the regression problem $y = f(\boldsymbol{x}) + \varepsilon$, assuming noise obeys a (usually normal) distribution, i.e., $\varepsilon \sim \mathcal{N}(0, \sigma_\varepsilon^2)$. The joint distribution of the observed values $\boldsymbol{y}$ and the predicted values $\boldsymbol{f}^{(*)}$ at the test points $\boldsymbol{x}^{(*)} = (\boldsymbol{x}^{(n+1)}, \cdots, \boldsymbol{x}^{(n+m)})^\intercal$ can be obtained:

$$\begin{bmatrix} \boldsymbol{y} \\ \boldsymbol{f}^{(*)} \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} m(\boldsymbol{x}) \\ m(\boldsymbol{x}^{(*)}) \end{bmatrix}, \begin{bmatrix} \boldsymbol{K}(\boldsymbol{x}, \boldsymbol{x}) + \sigma_n^2 \boldsymbol{I} & \boldsymbol{K}(\boldsymbol{x}, \boldsymbol{x}^{(*)}) \\ \boldsymbol{K}(\boldsymbol{x}^{(*)}, \boldsymbol{x}) & \boldsymbol{K}(\boldsymbol{x}^{(*)}, \boldsymbol{x}^{(*)}) \end{bmatrix} \right) \tag{2.24}$$

where the mean function is usually set to 0, i.e., $m(\boldsymbol{x}) = m(\boldsymbol{x}^{(*)}) = 0$, $\boldsymbol{K}(\boldsymbol{x}, \boldsymbol{x})$ is the $n \times n$ symmetric semi-positive definite covariance matrix; $\boldsymbol{K}(\boldsymbol{x}^{(*)}, \boldsymbol{x}) = \boldsymbol{K}(\boldsymbol{x}, \boldsymbol{x}^{(*)})^\intercal$ is an $m \times n$ matrix with $[\boldsymbol{K}(\boldsymbol{x}^{(*)}, \boldsymbol{x})]_{ij} = \boldsymbol{K}(\boldsymbol{x}^{(n+i)}, \boldsymbol{x}^{(j)})$; $\boldsymbol{K}(\boldsymbol{x}^{(*)}, \boldsymbol{x}^{(*)})$ is a matrix of order $m \times m$ and $[\boldsymbol{K}(\boldsymbol{x}^{(*)}, \boldsymbol{x}^{(*)})]_{ij} = \boldsymbol{K}(\boldsymbol{x}^{(n+i)}, \boldsymbol{x}^{(n+j)})$. This gives the posterior distribution of $\boldsymbol{f}^{(*)}$ the posterior distribution

$$\boldsymbol{f}^{(*)} \mid \boldsymbol{x}, \boldsymbol{x}^{(*)}, \boldsymbol{y} \sim \mathcal{N}\big(\overline{\boldsymbol{f}^{(*)}}, \operatorname{cov}(\boldsymbol{f}^{(*)})\big) \tag{2.25}$$

where

$$\begin{aligned}\overline{\boldsymbol{f}^{(*)}} &= \mathbb{E}[\boldsymbol{f}^{(*)} \mid \boldsymbol{x}, \boldsymbol{x}^{(*)}, \boldsymbol{y}] \\ &= m(\boldsymbol{x}^{(*)}) + \boldsymbol{K}(\boldsymbol{x}^{(*)}, \boldsymbol{x})[\boldsymbol{K}(\boldsymbol{x}, \boldsymbol{x}) + \sigma_\varepsilon^2 \boldsymbol{I}]^{-1}\big(\boldsymbol{y} - m(\boldsymbol{x})\big)\end{aligned} \tag{2.26}$$

and

$$\operatorname{cov}(\boldsymbol{f}^{(*)}) = \boldsymbol{K}(\boldsymbol{x}^{(*)}, \boldsymbol{x}^{(*)}) - \boldsymbol{K}(\boldsymbol{x}^{(*)}, \boldsymbol{x})[\boldsymbol{K}(\boldsymbol{x}, \boldsymbol{x}) + \sigma_\varepsilon^2 \boldsymbol{I}]^{-1}\boldsymbol{K}(\boldsymbol{x}, \boldsymbol{x}^{(*)}). \tag{2.27}$$

Gaussian process regression can capture different statistical features by choosing different covariance functions (kernel functions) [57], the most commonly used one is the Squared Exponential (SE) kernel:

$$K_{SE} = \eta^2 \exp\left(-\frac{(\boldsymbol{x} - \boldsymbol{x}')^2}{2l^2}\right) \tag{2.28}$$

where η^2 is the signal variance, l is the length scale. Hence, $\boldsymbol{\theta} = \{\eta^2, l, \sigma_n^2\}$ are the hyperparameters of the model. The optimal hyperparameters are usually obtained by maximizing the log marginal likelihood function [57].

Naïve Bayes

The naïve Bayes method is a classification technique that employs Bayes' theorem and the assumption of conditional independence of features to classify samples [21, 36], which is both easy to implement and effective in terms of learning and prediction. The naïve Bayesian method is a typical generative learning method, which learns the joint probability distribution $p(\boldsymbol{x}, y)$ from the training data, and then finds the posterior probability distribution $p(y \mid \boldsymbol{x})$. Specifically, the training data are utilized to learn the estimates of $p(\boldsymbol{x} \mid y)$ and $p(y)$ to obtain the joint probability distribution

$$p(\boldsymbol{x}, y) = p(y)\, p(\boldsymbol{x} \mid y) \tag{2.29}$$

and the probability estimation methods can be either maximum likelihood estimation or Bayesian estimation.

The basic assumption of the naïve Bayesian approach is conditional independence of features, which is a strong assumption:

$$p(\boldsymbol{x} \mid y = c_k) = p(x_1, \cdots, x_n \mid y = c_k) = \prod_{j=1}^{n} p(x_j \mid y = c_k). \tag{2.30}$$

Because of this assumption, the model contains fewer conditional probabilities, which greatly simplifies both learning and prediction. However, this approach may result in lower performance.

Naïve Bayesian uses Bayes' theorem with a learned joint probability model for classification prediction

$$p(y \mid \boldsymbol{x}) = \frac{p(\boldsymbol{x}, y)}{p(\boldsymbol{x})} = \frac{p(y)\, p(\boldsymbol{x} \mid y)}{\sum_y p(y)\, p(\boldsymbol{x} \mid y)} \tag{2.31}$$

and given an input $\boldsymbol{x}$, the output y with the highest posterior probability is obtained by

$$y = \arg\max_{c_k} p(y = c_k) \prod_{j=1}^{n} p(x_j \mid y = c_k). \tag{2.32}$$

Decision tree

Decision Tree (DT) is a kind of basic classification and regression method [58]. As an example, a classification DT model is a tree structure and represents the process of classifying instances based on features. It can be viewed as a set of if-then rules or a conditional probability distribution defined on the feature and class spaces. A DT consists of nodes and directed edges and there are two types of nodes: internal nodes and leaf nodes. An internal node represents a decision based on a particular feature, while a leaf node represents a class. A schematic diagram of this kind of model with a tree-like structure is shown in Figure 2.8.

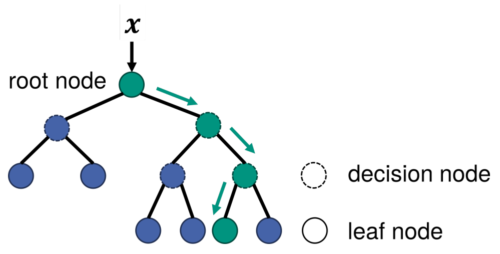


Figure 2.8: Schematic illustration of the decision tree.

The learning process of DT aims to construct a DT that fits the training data well and has low complexity. However, it is a Nondeterministic Polynomial-time complete (NP-complete) problem to select the optimal DT directly from all possible decision trees, hence heuristic methods are usually used to learn sub-optimal ones. The process of learning DT typically involves three steps: feature selection, tree generation, and tree pruning. The purpose of feature selection is to select features that are capable of classifying the training data and the key to feature selection is its selection criterion. The commonly used criteria are information gain (ID3) [58], information gain ratio (C4.5) [59], and Gini index (Classification And Regression Tree - CART) [60]. Decision trees are often generated recursively by calculating the criteria metrics, starting from the root node. This is equivalent to continuously selecting locally optimal features using the criteria or partitioning the training set into subsets that can be classified almost correctly [61].

However, this process tends to generate a DT that suffers from an overfitting problem, hence it needs to be pruned to simplify the learned DT [36]. The pruning of a DT often works by cutting off some leaf nodes or sub-trees above the original leaf nodes from the generated tree and taking its parent or root node as the new leaf node.

Ensemble models

Ensemble learning [62] is a class of very popular machine learning methods and it is not a standalone machine learning algorithm. Instead, it works by constructing and combining multiple machine learners to complete the task, which can be interpreted as "learning from many". For the training set data, one can end up with a good-performing learning model by training a number of individual machine learning models (called learners, which are usually weak learners with mediocre performance compared to random guessing), which can be combined with certain strategies into a better-performing model (usually named as strong learner) [63]. There are two main problems to be solved in ensemble learning, the first one is how to get several individual learners and the second one is how to choose a combining strategy to aggregate these individual learners into a final model.

For the first issue, how to get a group of individual learners, homogeneous learners (all the individual learners are of the same kind) are more widely used, with CART decision trees and neural networks being the most common ones (while heterogeneous learners, being of different kinds, are less used). Homogeneous learners can be further subdivided into two groups based on the presence of a dependency among individual learners. The first group comprises learners with a strong inter-dependency, where individual learners are generated sequentially and the representative algorithms used are boosting algorithms [64]. The second group is that there is little interdependence between individual learners, thereby enabling the parallel creation of a batch of individual learners, with bagging algorithms serving as the representative algorithms [65].

The second issue concerns the aggregation strategy, specifically, how to combine the prediction of the individual learners as the final result. The most widely used strategies are: (i) the averaging method for regression problems; (ii) the majority voting method for classification problems and (iii) the learning method for both problems. One of the most representative learning methods is stacking [66], that is, instead of carrying out simple processing of the results of the weak learners, a layer of additional learners is added. In other words, some weak learners (referred to as the primary learner) are trained on the samples first, and their outputs are used as inputs to train a second layer of new learners (referred to as the secondary learner). To obtain the final result, the primary learner first makes predictions on samples to generate outputs which are then used as inputs for the secondary learner to perform prediction.

Boosting

As shown in Figure 2.9a, the boosting algorithm [64] operates by initially training a first weak learner from the original training set with initial weights. The weights of the training samples are then updated based on the first weak learner's results. The weights of the training samples that have a high error rate will be increased, in this way, these points can receive increased attention in the training of a second weak learner. The above process is iterated until the count of weak learners equals the pre-specified value of T. Subsequently, these T weak learners are combined by certain ensemble strategies to build the final strong learner. The most well-known algorithms in the boosting family are the AdaBoost algorithm [67] and the Boosting tree families [68], the most widely used being the Gradient Boosting Decision Tree (GBDT) [69] and XGBoost [70].

Bagging

The bagging algorithm [65] is different from boosting in that its weak learners are independent, allowing for parallel generation. This principle is illustrated in Figure 2.9b. The training set for building each weak learner in bagging is acquired through random sampling. A total of T sampling sets are obtained by randomly sampling T times, and T weak learners are trained independently and separately based on each sampling set. In the end, the T weak learners are combined to form the final, stronger learner.

For random sampling, the Bootstrap method [71] is commonly adopted. This method works by generating T subsets of samples by sampling the original data set with replacement and using these subsets to train T different models. Due to random sampling, each sample set is distinct from the original training set and from other sample sets, enabling the generation of multiple different weak learners.

The random forest model [72, 73] is one of the most popular bagging models and it is a specialized and advanced version compared to the original idea of bagging. It uses decision trees as weak learners, and it introduces additional randomness during the training process by randomly selecting subsets of features to further improve performance. However, the basic idea of it has not deviated too far from the scope of bagging.

A schematic comparison of these two principles, boosting and bagging, is shown in Figure 2.9.

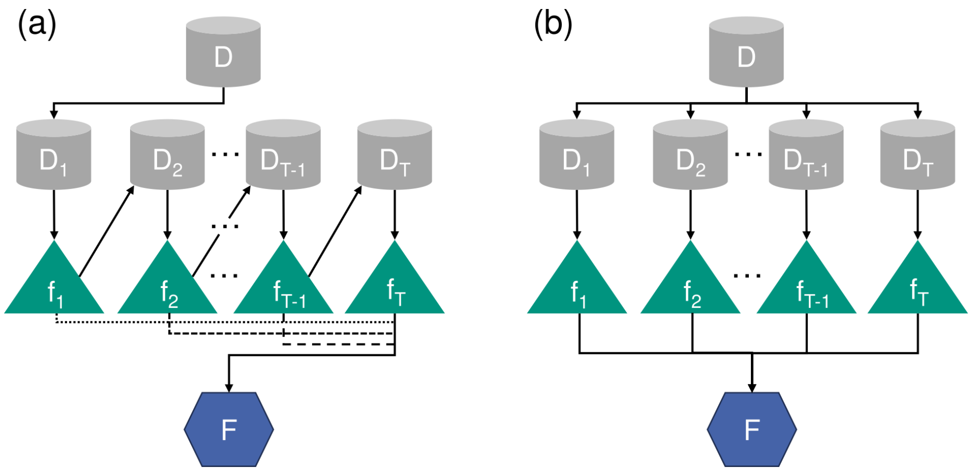


Figure 2.9: Schematic comparison of (a) boosting and (b) bagging. The original data set D is formed into different (sequential or parallel, respectively) subsets $\{D_1, D_2, \cdots D_T\}$ according to the strategies of different methods. The learned T weak learners (models) are noted as $\{f_1, f_2, \cdots, f_T\}$ and the ensemble strong learner is noted as F.

2.5.2 Unsupervised learning

Unsupervised learning [36, 35, 21, 74] refers to the automatic learning of valuable information from training samples that do not contain labels. Typically it includes clustering, dimensionality reduction, and probability estimation. Unsupervised learning is useful for data analysis and can be used as preprocessing for supervised learning.

The goal of unsupervised learning is to identify statistical patterns and the underlying structure within a data set. The fundamental concept of unsupervised learning involves compressing the data (typically in matrix form) to uncover this structure, with the assumption that the compressed version resulting in the least loss contains the most essential structure. Assuming that $\mathcal{X}$ is the input space and $\mathcal{Z}$ is the implicit structure space, the model to be learned can be represented as a function $\boldsymbol{z} = g(\boldsymbol{x})$, a conditional probability distribution $p(\boldsymbol{z} \mid \boldsymbol{x})$, or $p(\boldsymbol{x} \mid \boldsymbol{z})$, where $\boldsymbol{x} \in \mathcal{X}$ is the input and $\boldsymbol{z} \in \mathcal{Z}$ is the output. The set of all potential models is known as the hypothesis space. The objective of unsupervised learning is to pick out the best model from the hypothesis space based on a specific evaluation criterion.

Similar to supervised learning, unsupervised learning also consists of the learning system and the prediction system, as is shown in Figure 2.10. In the learning process, the learning system tries to learn from the training data set to obtain an optimal model. In the prediction process, the prediction system performs either clustering or dimensionality reduction for a given input $\boldsymbol{x}^{(N+1)}$ by yielding the corresponding output $\boldsymbol{z}^{(N+1)}$ by the model $\boldsymbol{z}^{(N+1)} = g(\boldsymbol{x}^{(N+1)})$ or $\boldsymbol{z}^{(N+1)} = \arg\max_{\boldsymbol{z}} p(\boldsymbol{z} \mid \boldsymbol{x}^{(N+1)})$. For the probability estimation task, it predicts the result $p(\boldsymbol{x}^{(N+1)} \mid \boldsymbol{z}^{(N+1)})$ with the model $p(\boldsymbol{x} \mid \boldsymbol{z})$.

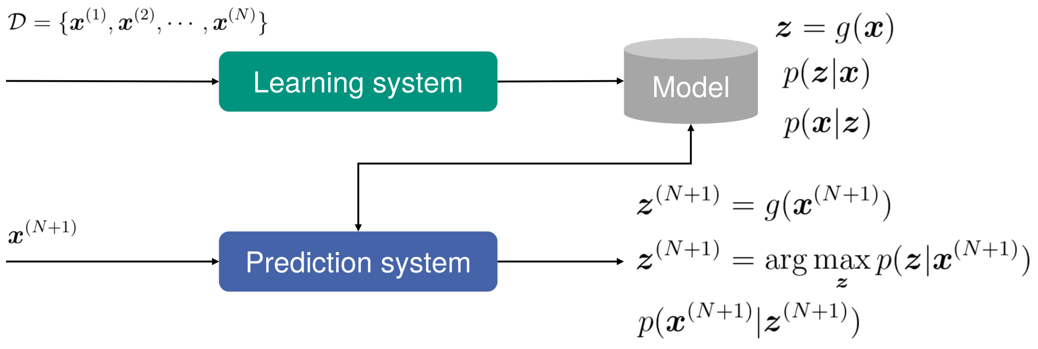

$\mathcal{D} = \{\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \cdots, \boldsymbol{x}^{(N)}\}$
Learning system
Model
$\boldsymbol{z} = g(\boldsymbol{x})$
$p(\boldsymbol{z}|\boldsymbol{x})$
$p(\boldsymbol{x}|\boldsymbol{z})$
$\boldsymbol{x}^{(N+1)}$
Prediction system
$\boldsymbol{z}^{(N+1)} = g(\boldsymbol{x}^{(N+1)})$
$\boldsymbol{z}^{(N+1)} = \arg\max_{\boldsymbol{z}} p(\boldsymbol{z}|\boldsymbol{x}^{(N+1)})$
$p(\boldsymbol{x}^{(N+1)}|\boldsymbol{z}^{(N+1)})$

Figure 2.10: Schematic overview of unsupervised learning.

Clustering

Clustering [75] is a data analysis problem that involves grouping samples into "classes" or "clusters" based on the similarity or distance of their features. Intuitively, similar samples should be grouped in the same cluster while dissimilar samples should be classified into different clusters. Clustering involves automatically discovering the sample groups from the data, without prior knowledge of the sample classes. Typically, the number of sample clusters is predetermined. If a sample can solely belong to one class, it is referred to as hard clustering. If a sample can belong to multiple classes, then it is referred to as soft clustering.

There are various definitions of similarity or distance, and the selection depends on the problem's characteristics. The fundamental problem in clustering is the selection of an appropriate similarity or distance measure since it directly affects the results. Commonly used distance measures (smaller values indicate better similarity) are Minkowski distances, including Euclidean distance, Manhattan distance, Chebyshev distance, and Mahalanobis distance, while commonly adopted similarity measures (larger values indicate better similarity) are correlation coefficient and cosine of the angle, and so on [74].

Suppose the input space is the Euclidean space $\mathcal{X} \subseteq \mathbb{R}^D$, and the output space is the set of categories $\mathcal{Z} = \{1, 2, \cdots, k\}$. Clustering is modeled by the function $\boldsymbol{z} = g_{\boldsymbol{\theta}}(\boldsymbol{x})$ (hard clustering model) or the conditional probability distribution $p_{\boldsymbol{\theta}}(\boldsymbol{z} \mid \boldsymbol{x})$ (soft clustering model), where $\boldsymbol{x} \in \mathcal{X}$ is a vector feature of sample, $\boldsymbol{z} \in \mathcal{Z}$ is the category of samples, and $\boldsymbol{\theta}$ are the parameters. There are numerous clustering algorithms in the literature, however, proposing a concise categorization of clustering methods is challenging due to potential category overlaps. Generally, the basic clustering algorithms can be classified as the following.

Partitioning methods divide the data with a set of N objects into k clusters $k \leqslant N$.

This allows each group to contain at least one object, and each cluster represents a particular partition. Most partitioning methods rely on distance measurements. To construct k partitions, the initial step is to create an initial partition which is then refined using iterative relocation to optimize the partition. A crucial requirement for an efficient partition is that objects within the same group are as similar as possible, while objects in different groups are as dissimilar as possible. To obtain the best possible outcome, clustering based on partition may necessitate examining all possible partitions, which can be computationally intensive. In reality, most popular algorithms use heuristics to asymptotically improve the quality of clustering and approximate the local optimal solution, such as k-means and k-center-point algorithms [76].

The k-means clustering [76] is a clustering algorithm that partitions a collection of samples and divides the sample set into k subsets, creating k classes. Each sample is assigned to the class with the nearest center, based on its distance from its corresponding center. The k-means clustering algorithm follows an iterative process, with two steps in each iteration. Initially, the algorithm selects the centers of the k classes and assigns each sample one by one to the class whose center is nearest, producing intermediate clustering results. Afterward, the mean of the samples in each class is updated, which is then regarded as the new center of that class. These steps are repeated until convergence. It should be noted that k-means clustering is a heuristic approach that cannot guarantee convergence to the global optimum, and the choice of the initial center directly affects the clustering results.

Hierarchical methods create a hierarchical decomposition of a given set of data objects [77]. Depending on the formation of the hierarchical decomposition, hierarchical methods can be classified as either agglomerative or divisive. Agglomerative methods, also known as bottom-up methods, start with each object as a separate group and then merge similar objects or groups one by one until all combinations are merged into a single group (the top level of the hierarchy), or until a termination condition is met. The divisive approach, also known as the top-down approach, begins by placing all objects in a cluster, and in each successive iteration, a cluster is divided into smaller clusters until eventually each object is in a separate cluster, or a termination condition is met.

Density-based methods have been developed to solve the problem of difficulty in detecting clusters of irregular shapes for most distance-based partitioning methods. The main idea is to continue growing a given cluster as long as the density (number of objects or data points) in the "neighborhood" exceeds a certain threshold. That is, for each data point in a given cluster, a neighborhood of a given radius must contain at least a minimum number of objects. This approach effectively filters out noise and outlier points while also identifying clusters with varying shapes. Density-based methods have the capability to partition a group of objects into distinct clusters that are either hierarchical in structure or mutually exclusive. It is noteworthy that, in general, density-based methods only consider mutually exclusive clusters and do not consider fuzzy clusters.

Density-Based Spatial Clustering of Applications with Noise (DBSCAN) [78] is an exemplary method that performs better with noisy data. This approach can be compared to drawing a circle to get a cluster, in which two parameters need to be defined: the maximum radius of the circle and the minimum number of points that should be contained in the circle. Any data point located within the circle is considered a member of the same class. However, setting the parameters can be problematic, especially given their high sensitivity to configuration. An extension of DBSCAN called Ordering Points To Identify Clustering Structure (OPTICS) [79] enhances DBSCAN by prioritizing the discovery of high-density regions and adjusting the parameters based on their features, thereby addressing DBSCAN's weaknesses.

Grid-based methods quantize the object space into a finite number of cells, forming a grid structure. All clustering operations are performed on this grid structure (i.e., the quantized space). The primary benefit of this approach is its speed, as the time consumption is typically determined only by the number of dimensions of the quantized space and not by the number of data objects. STING (STatistical INformation Grid) [80] and CLIQUE (CLustering In QUEst) [81] represent typical algorithms within this category of methodologies.

Model-based methods rely heavily on probabilistic and neural network models, with a major emphasis on probabilistic models. Probabilistic models, particularly generative models, are used to group data from the same "class" in the same probability distribution. These methods have the advantage of providing a flexible way to define "classes" probabilistically, expressing the characteristics of each class via parameters. Nonetheless, a downside is their potential lack of efficiency when dealing with numerous distributions and limited data.

One of the most common and widely employed techniques is Gaussian Mixture Models (GMM) [82]. Regarding strategies based on neural network models, they mainly pertain to Self-Organizing Maps (SOM) [83]. This algorithm assumes the existence of topological structure or order among input objects, facilitating a mapping that decreases dimensionality from higher dimensional input space to the 2-dimensional output plane while preserving topological properties.

Dimensionality reduction

Dimensionality reduction [36] involves transforming data from a high-dimensional space to a low-dimensional one. This approach enables a better representation of the structure and relationships between samples. It assumes that the original data existed in a low-dimensional space or approximately the same way. A high-dimensional space is generally a high-dimensional Euclidean space, whereas a low-dimensional space is either a low-dimensional Euclidean space or a manifold [34]. Low-dimensional spaces are not predetermined but rather derived from data, with their dimensions often specified in advance. During the process of dimensionality reduction, it is crucial to minimize the loss of information in the sample when reducing from higher to lower dimensions.

Suppose te $\mathcal{X} \subseteq \mathbb{R}^D$, and the output space is also the Euclidean space $\mathcal{Z} \subseteq \mathbb{R}^{D'}$, $D' \ll D$, the latter being of lower dimension than the former. The model for dimensionality reduction is the function $\boldsymbol{z} = g_{\boldsymbol{\theta}}(\boldsymbol{x})$, where $\boldsymbol{x} \in \mathcal{X}$ is the high-dimensional feature vector of the sample, $\boldsymbol{z} \in \mathcal{Z}$ is the low-dimensional feature vector of the sample, $\boldsymbol{\theta}$ are the parameters, and the function can be linear or nonlinear. The process of dimensionality reduction is the process of learning a dimensionality reduction model, and each sample is converted from a high dimensional vector to a low dimensional vector $\boldsymbol{z}^{(n)} = g_{\boldsymbol{\theta}}(\boldsymbol{x}^{(n)}), n = 1, 2, \cdots, N$.

Principal Component Analysis (PCA) [44, 45] is a frequently employed technique for reducing data dimensionality with the idea of maximizing variance in the transformed space. For example, as shown in Figure 2.11, when projecting two-dimensional data into a one-dimensional space, the direction with the greatest variance is chosen to maximize the variance of the data and to preserve more information about the original data. PCA is an unsupervised learning method that can be used as a data preprocessing method for supervised learning to remove noise and reduce the correlation between features.

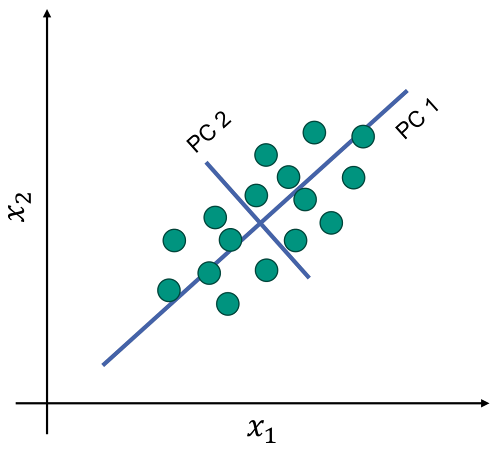


Figure 2.11: Schematic illustration of PCA applied to two-dimensional data.

Suppose there is a set of D-dimensional samples $\boldsymbol{x}^{(n)} \in \mathbb{R}^D$,$1 \leq n \leq N$, which are to be projected into a one-dimensional space with projection vectors $\boldsymbol{w} \in \mathbb{R}^D$. Without loss of generality, $\boldsymbol{w}$ is restricted to be of mode 1, i.e., $\boldsymbol{w}^{\mathrm{T}}\boldsymbol{w} = 1$. Each sample point $\boldsymbol{x}^{(n)}$ after projection is represented as $\boldsymbol{z}^{(n)} = \boldsymbol{w}^\intercal \boldsymbol{x}^{(n)}$. Using the matrix $\boldsymbol{X} = [\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \cdots, \boldsymbol{x}^{(N)}]$ to represent the input samples, and $\bar{\boldsymbol{x}} = \frac{1}{N}\sum_{n=1}^{N} \boldsymbol{x}^{(n)}$ to be the centroid of the original samples, the variance of all the samples after projection is

$$\begin{aligned}\sigma(\boldsymbol{X};\boldsymbol{w}) &= \frac{1}{N}\sum_{n=1}^{N}(\boldsymbol{w}^\intercal \boldsymbol{x}^{(n)} - \boldsymbol{w}^\intercal \bar{\boldsymbol{x}})^2 \\ &= \frac{1}{N}(\boldsymbol{w}^\intercal \boldsymbol{X} - \boldsymbol{w}^\intercal \bar{\boldsymbol{X}})(\boldsymbol{w}^\intercal \boldsymbol{X} - \boldsymbol{w}^\intercal \bar{\boldsymbol{X}})^\intercal \\ &= \boldsymbol{w}^\intercal \boldsymbol{\Sigma} \boldsymbol{w}\end{aligned} \tag{2.33}$$

where $\bar{\boldsymbol{X}} = \bar{\boldsymbol{x}}\mathbf{1}_D^\intercal$ is the cross product of the vector $\bar{\boldsymbol{x}}$ and the D-dimensional all-1 vector $\mathbf{1}_D$, i.e., the matrix consisting of D columns $\bar{\boldsymbol{x}}$, and $\boldsymbol{\Sigma} = \frac{1}{N}(\boldsymbol{X} - \bar{\boldsymbol{X}})(\boldsymbol{X} - \bar{\boldsymbol{X}})^\intercal$ is the covariance matrix of the original samples.

It is required to maximize the projected variance $\sigma(\boldsymbol{X};\boldsymbol{w})$ while satisfying the condition $\boldsymbol{w}^\intercal \boldsymbol{w} = 1$, hence this problem can be converted to an unconstrained optimization problem using the Lagrange method:

$$\max_{\boldsymbol{w}} \boldsymbol{w}^\intercal \boldsymbol{\Sigma} \boldsymbol{w} + \lambda(1 - \boldsymbol{w}^\intercal \boldsymbol{w}) \tag{2.34}$$

where λ is the Lagrange multiplier. The solution yields $\boldsymbol{\Sigma}\boldsymbol{w} = \lambda \boldsymbol{w}$. From the above, it is known that $\boldsymbol{w}$ is the eigenvector of the covariance matrix $\boldsymbol{\Sigma}$, and λ is the eigenvalue. At the same time,

$$\sigma(\boldsymbol{X};\boldsymbol{w}) = \boldsymbol{w}^\intercal \boldsymbol{\Sigma} \boldsymbol{w} = \boldsymbol{w}^\intercal \lambda \boldsymbol{w} = \lambda. \tag{2.35}$$

λ is also the variance of the sample after projection. Thus, principal component analysis can be converted into a matrix eigenvalue decomposition problem where the projection vector $\boldsymbol{w}$ is the eigenvector corresponding to the largest eigenvalue of the matrix $\boldsymbol{\Sigma}$.

If it is desired to project the samples into the D'-dimensional space by the projection matrix $\boldsymbol{W} \in \mathbb{R}^{D \times D'}$, and the projection matrix satisfies $\boldsymbol{W}^\intercal \boldsymbol{W} = \boldsymbol{I}$ as an identity matrix, it only needs to arrange the eigenvalues of $\boldsymbol{\Sigma}$ from the largest to the smallest and keep the first D' eigenvectors, and the corresponding eigenvectors are the optimal projection matrix:

$$\boldsymbol{\Sigma}\boldsymbol{W} = \boldsymbol{W}\,\mathrm{diag}(\boldsymbol{\lambda}) \tag{2.36}$$

where diag is the diagonal matrix with $\boldsymbol{\lambda} = [\lambda_1, \cdots, \lambda_{D'}]$ are the leading D' largest eigenvalues.

The principal components transformation can also be associated with another matrix factorization, the Singular Value Decomposition (SVD) of a data matrix $\boldsymbol{X}$, as there are many efficient algorithms to calculate the SVD, hence computing the SVD is now the standard way to calculate the PCA [84].

Manifold learning is a technique for reducing dimensionality that is inspired by the concept of topological manifolds [34]. A “manifold” is a space that is locally homomorphic to Euclidean space, which implies that it has Euclidean properties in its local regions and enables distance calculations using Euclidean distances. Such an idea has a significant impact on dimensionality reduction techniques: if a low-dimensional manifold is embedded in a high-dimensional space, the distribution of data samples in that space may appear complex, but Euclidean characteristics are still preserved in local regions. Therefore, it is feasible to create local dimensionality reduction mappings and extend them globally. By reducing dimensionality to two or three dimensions, the data can be easily visualized, which makes manifold learning a useful tool for visualization.

One of its primary usages is to perform nonlinear dimensionality reduction. This process takes into account not only distance but also the topological structure that generates the data. Additionally, manifolds have the capability to capture the essence of data. In simpler terms, this process involves using a mapping that is able to reduce data from a high-dimensional space to a low-dimensional space without any loss of information. This mapping takes the original data as input and outputs more intrinsic features of the data, similar to compression where a smaller amount of data can represent the original data as accurately as possible.

Commonly seen methods are Isometric mapping (Isomap) [85], Locally Linear Embedding (LLE) [86], Laplacian Eigenmaps (LE) [87], Stochastic Neighbor Embedding (SNE) as well as t-SNE [88, 89], and Uniform Manifold Approximation and Projection (UMAP) [90].

Non-negative Matrix Factorization (NMF) is a common method of matrix factorization which can be used for dimension reduction [91, 92]. Its purpose is factoring a matrix $\boldsymbol{X}$ with non-negative entries into two matrices $\boldsymbol{W}$ and $\boldsymbol{H}$ that also have only non-negative entries, which are typically required to have a much lower rank than the original matrix. That is, the goal of NMF is to achieve $\boldsymbol{X} \simeq \boldsymbol{W}\boldsymbol{H}$ by finding $\boldsymbol{W}$ and $\boldsymbol{H}$ that minimize the error function (using the Frobenius norm) $||\boldsymbol{X} - \boldsymbol{W}\boldsymbol{H}||_{\text{fro}}$, subject to $\boldsymbol{W} \geq 0, \boldsymbol{H} \geq 0$. The non-negativity constraints can lead to a decomposition that allows for a semantically meaningful interpretation of the coefficients. However, in most cases, the resulting factorization problem has no exact solution, thus requiring optimization procedures for finding suitable numerical approximations. The commonly used way to solve this problem is the iterative methods, such as the gradient descent method or quasi-Newton’s method.

2.5.3 Semi-supervised learning and active learning

Semi-supervised learning and active learning are closer to supervised learning. Semi-supervised learning refers to a machine learning problem that involves learning predictive models using both labeled and unlabeled data. In practice, the amount of labeled data is usually small, while the amount of unlabeled data is large. This is because labeling data usually requires significant manual labor, whereas collecting unlabeled data is easier. Semi-supervised learning aims to utilize the unlabeled data to support the labeled data, resulting in enhanced learning outcomes at a reduced cost.

Active learning is a machine learning task in which the machine continuously and actively selects samples for labeling and the labeled samples are then used to develop a predictive model. The main objective of active learning is to train a good-performing model at a lower labeling cost by identifying the most useful and valuable instances (samples), in other words, it is a subfield of machine learning that investigates the optimal allocation strategy under finite resources.

Bayesian optimization-based active learning

The goal of active learning is to minimize the number of attempts needed to attain a performance that is as good as or better than that achieved by exhaustive exploration. Bayesian Optimization (BO) is one of the most commonly used active learning strategies and it works by balancing the trade-offs between exploring and exploiting. BO is a class of optimization methods that focuses on solving the problem $\max_{\boldsymbol{x} \in \mathcal{X}} f(\boldsymbol{x})$ within an input domain $\mathcal{X} \subseteq \mathbb{R}^D$ as the bounding box. The ability to optimize the expensive black-box derivative-free functions makes it extremely versatile [93]. Recently, it has become popular for tuning hyperparameters in ML models, especially deep neural networks [94].

A typical BO method involves two primary components:

1. a surrogate model for statistical inference, typically Gaussian Process (GP) [95, 57], which is a widely used surrogate for modeling objective functions. The function f is typically assumed to be a GP which is determined by a mean function μ and a covariance kernel K, $f \sim \mathcal{GP}(\mu, K)$;

2. an acquisition function $a(\boldsymbol{x})$ that decides where to sample the next most promising candidate with the largest objective value. The acquisition function is usually an inexpensive function that defines a balance between exploring new areas in the objective space and exploiting areas that are already known to have favorable values [96]. Typical acquisition functions are the Probability of Improvement (PI) [97], Expected Improvement (EI) [98, 99], and Upper Confidence Bound (UCB) [100]. These functions are different in how the trade-off is made between exploration and

exploitation. Such a strategy is important for finding the global optimum efficiently instead of being trapped in a local optimum.

Given the observed data set $\mathcal{D}$, the question would be where the next point to observe the function is. The meta-approach in BO is to design an acquisition function $a(\boldsymbol{x})$. The acquisition function is usually an inexpensive function, which defines a balance between exploring new areas in the objective space and exploiting areas that are already known to have favorable values [96]. This strategy is important for helping to find the global optimum efficiently instead of being trapped in a local optimum.

The PI is one of the earliest acquisition functions designed for BO which suggests maximizing the probability of improvement (P) over the current best-observed value $f(\boldsymbol{x}^+)$, where $\boldsymbol{x}^+ = \arg\max_{\boldsymbol{x}\in\mathcal{D}_{1:t}} f(\boldsymbol{x})$ with the observed data set $\mathcal{D}_{1:t}$, so that

$$\boldsymbol{x}_{t+1} = \arg\max_{\boldsymbol{x}\in\mathcal{X}} P\big(f(\boldsymbol{x}) \geq f(\boldsymbol{x}^+)\big) = \arg\max_{\boldsymbol{x}\in\mathcal{X}} \Phi\left(\frac{\mu_t(\boldsymbol{x};\mathcal{D}_{1:t}) - f(\boldsymbol{x}^+)}{\sigma_t(\boldsymbol{x};\mathcal{D}_{1:t})}\right) \tag{2.37}$$

where $\Phi(\cdot)$ is the normal Cumulative Distribution Function (CDF), μ_t and σ_t are the posterior mean and posterior standard deviation at iteration t.

Alternatively, maximizing EI over the current best value can also be chosen, which accounts for the amount of the improvement (while PI does not). It can be computed analytically as:

$$\boldsymbol{x}_{t+1} = \arg\max_{\boldsymbol{x}\in\mathcal{X}} \Big[\big(\mu_t(\boldsymbol{x};\mathcal{D}_{1:t}) - f(\boldsymbol{x}^+)\big)\Phi(Z) + \sigma_t(\boldsymbol{x};\mathcal{D}_{1:t})\phi(Z)\Big] \tag{2.38}$$

with

$$Z = \frac{\mu_t(\boldsymbol{x};\mathcal{D}_{1:t}) - f(\boldsymbol{x}^+)}{\sigma_t(\boldsymbol{x};\mathcal{D}_{1:t})}$$

where $\Phi(\cdot)$ and $\phi(\cdot)$ are the CDF and Probability Density Function (PDF) of the standard normal distribution, respectively.

The acquisition function of UCB takes the form:

$$\boldsymbol{x}_{t+1} = \arg\max_{\boldsymbol{x}\in\mathcal{X}} \left[\mu_t(\boldsymbol{x};\mathcal{D}_{1:t}) + \kappa\sigma_t(\boldsymbol{x};\mathcal{D}_{1:t})\right]. \tag{2.39}$$

This function can be intuitively interpreted as a weighted sum of the prediction of $f(\boldsymbol{x})$ and its uncertainty. κ is a tunable hyperparameter that controls how much of the variance in the predicted values should be taken into account, where higher value favors exploration over exploitation and vice versa [57].

To find the global optimum in an effective way, the search strategy needs to follow the aforementioned principle, i.e., combining exploration and exploitation. The model should not only focus on the local region where the known maximum value is located but also explore the whole search space wisely.

In each iteration of BO, the surrogate model is fitted based on the known samples to obtain information on the statistical mean estimations and uncertainties (represented in variances) across the input domain space. The information is then fed into the selected acquisition function to decide the next design points that may potentially improve the performance of the target function at best. New validations will be conducted on these new candidates and their results will serve as new inputs to update the whole model for the next iteration. This loop will continue until certain criteria are met, such as the achievement of sufficient performance of the original function. On this account, the merit is that expensive evaluations only need to be conducted on these promising candidates, which has the potential to reduce the number of required evaluations. Following this strategy, the overall computational cost of the design process is reduced considerably. Besides, it should be noted that in practice BO is usually limited to data sets with a typical size of $\sim$10k samples due to its prohibitive computational cost when dealing with large data sets [57, 101].

2.5.4 Reinforcement learning

Reinforcement Learning (RL) is a type of machine learning algorithm that achieves learning through interaction, which challenges an intelligent system to learn optimal behavioral strategies continuously while interacting with the environment. The interaction between the intelligent system and its environment is usually assumed to follow the Markov decision process [102]. The intelligent system can solely observe the data sequence resulting from this interaction. The aim of RL is to acquire the optimal sequential decision (policy). Compared to supervised learning, RL is an online learning mechanism that does not necessitate the explicit provision of training samples in the form of “input/output pairs”.

Figure 2.12 illustrates the interaction of the intelligent system with the environment. The intelligent system observes a state s_t with a reward r_t from the environment at each step t and then performs an action a_t. Based on the intelligent system’s decision, the environment outputs the next state s_{t+1} with a reward r_{t+1}. The policy being learned is based on the action taken in a given state. The objective of an intelligent system is to maximize long-term cumulative rewards rather than short-term rewards. The RL process involves the system continuously learning the optimal strategy through trial and error by interacting with the environment.

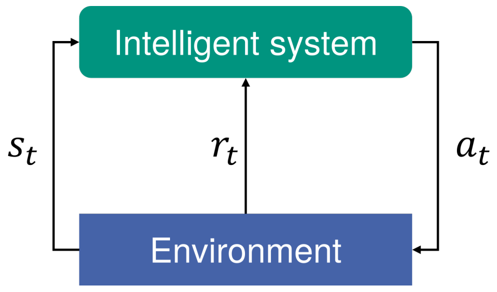


Figure 2.12: Illustration of the interaction between intelligent system and the environment.

The Markov decision process for RL is a stochastic process on a sequence of states, rewards, and actions, consisting of the quintuple $\langle \mathcal{S}, \mathcal{A}, p, r, \gamma \rangle$:

- $\mathcal{S}$ is a finite set of states;
- $\mathcal{A}$ is a finite set of actions;
- p is the transition probability function from state s to s' by taking action a: $p(s' \mid s, a) = p(s_{t+1} = s' \mid s_t = s, a_t = a)$;
- r is the reward function determined by an environment E: $r(s, a) = E(r_{t+1} \mid s_t = s, a_t = a)$;
- γ is discount factor: $\gamma \in [0, 1]$.

The Markov decision process has the property of Markovianity [103], whereby the succeeding state hinges solely on the previous state conditioned on the action applied, represented by a state transition probability function $p(s' \mid s, a)$. The obtained reward rests on the prior state tied to the action applied, denoted by the reward function $r(s, a)$.

The policy π is defined as a function $a = f(s)$ of the action in a given state or a conditional probability distribution $p(a \mid s)$. Given a policy π, the behavior of the intelligent system interacting with the environment is determined (either deterministically or stochastically).

The value function or state value function is defined as the mathematical expectation of the long-term cumulative reward of the strategy π starting from a certain state s:

$$v_\pi(s) = \mathbb{E}_\pi \left[r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots \mid s_t = s \right]. \tag{2.40}$$

The action value function is defined as the mathematical expectation of the long-term cumulative reward of a strategy π starting from a certain state s and action a:

$$q_\pi(s, a) = \mathbb{E}_\pi \left[r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots \mid s_t = s, a_t = a \right]. \tag{2.41}$$

The objective of RL is to choose the strategy with the highest value function π^* out of all available options. Typically, learning begins with a certain strategy and then proceeds to continuously enhance the current strategy. In this context, γ refers to how future rewards will decay.

RL methods include policy-based and value-based methods, which are model-free. Model-free, policy-based methods aim to determine the optimal policy, denoted by a function $a = f^*(s)$ or a conditional probability distribution $p^*(a \mid s)$. This approach enables optimal decision-making in the environment. The learning process typically begins with a particular strategy and advances by seeking a superior one. Model-free, value-based methods try to determine the optimal value function, specifically the optimal action value function $q^*(s, a)$. This approach enables the indirect acquisition of the optimal policy for selecting the appropriate action in a given state. Learning usually starts with a specific value function and proceeds by searching for a more optimal value function.

On the contrary, there are also model-based methods. Model-based approaches attempt to directly learn a model of the Markov decision process, including the transition probability function $p(s' \mid s, a)$ and the reward function $r(s, a)$. This allows the model to predict feedback from the environment and find the strategy π^* that maximizes the value function.

Deep reinforcement learning

In RL, it is generally necessary to model the policy $\pi(a \mid s)$, value function $v_\pi(s)$, and the action value function $q_\pi(s, a)$. Early RL algorithms focused on problems with discrete and finite states and actions, where tables could be used to record probabilities. Nevertheless, in many real-world problems, some tasks have a very large number of states and actions. For instance, the game of Go comprises numerous states. Additionally, there are tasks in which the states and actions are generally continuous, e.g., in autonomous driving, the perceived state of the environment includes various sensor data, such as vehicle speed and steering wheel angle. To address these challenges, a more robust policy function, such as a (deep) neural network, can be developed. This will enable the intelligent system to handle complex environments, enhance policy learning, and improve generalization capabilities. Deep Reinforcement Learning (DRL) [103] combines reinforcement learning and deep learning. Reinforcement learning is used to define and optimize the objective, while deep learning is utilized to model the policy and value functions. DRL is believed to have a degree of generalized intelligence to solve complex problems and has been very successful in many tasks.

2.6 Deep learning and artificial neural networks

Deep Learning (DL) [34, 22] typically employs complex models, wherein data flows through multiple linear or non-linear components of the model between input and target output, is a rapidly growing field of research in recent years, and has achieved great success in many fields. Since each component processes information and affects the subsequent components, it is unclear how much each contributes to the final output. This issue is known as the Credit Assignment Problem (CAP) [104]. In DL, CAP is a crucial challenge closely tied to parameter learning in each component. Artificial Neural Network (ANN), a mathematical model inspired by the workings of the human brain's nervous system, is currently the most effective solution to the CAP. The nervous system of the human brain is capable of encoding auditory and visual signals through several layers, abstracting and processing the signals from their low-level features, and ultimately acquiring the semantic representation of the original signals [34]. Similar to this, the ANN comprises artificial neurons and connections between neurons, which can be regarded as an information processing system from input to output. The complexity of the neural network model results in a longer transfer path from input to output, making the learning of complex neural networks a form of deep machine learning, commonly referred to as deep learning. ANN models are the most prevalent models used in DL.

The connections between individual nodes in ANN are given different weights that indicate how much one node affects another. Each node stands for a specific function, and data from other nodes is synthesized with their corresponding weights before being processed by an activation function, resulting in a new value of activity (either excitation or inhibition). From a systemic perspective, an ANN is a dynamic, nonlinear, and adaptable system consisting of a vast amount of neurons with complex and highly developed connections. Although the creation of an ANN is achievable, making it capable of learning poses considerable difficulty. Perceptrons [105] were the first neural networks with machine learning ideas, but their learning methods could not be extended to multi-layer neural networks and it was not until around 1980 that the backpropagation algorithm [106, 107] effectively solved the problem of multi-layer neural network learning and became the most popular learning algorithm for neural network models. Since an ANN can be used as a general function approximator (a two-layer neural network can approximate any function) [108, 109], it can be considered as a learnable function and applied to machine learning. The network capacity of ANN refers to its ability to shape complex functions, which is directly related to the amount and complexity of data it can store.

A diverse range of neural network architectures have been developed so far. The most commonly used three neural network structures are feedforward networks, memory networks, and graph networks.

Feedforward networks have neurons that are grouped based on the order of information reception, with each group serving as a neural layer. Neurons in each layer receive the outputs from neurons in the previous layer and transmit them to the next. The entire network's information is propagated solely in one direction, represented by a directed acyclic graph with no backward information flow. Feedforward networks consist of fully connected feedforward networks and convolutional neural networks. A feedforward network functions as a complex mapping from input space to output space through the combination of multiple simpler nonlinear functions.

Memory networks Memory networks, also referred to as feedback networks, are networks where neurons receive information from not only other neurons but also their own history. Unlike feedforward networks, neurons in a memory network possess a memory function and exhibit different states at different moments in time. Information propagation in a memory neural network can be either unidirectional or bidirectional, which can be represented as either a directed or undirected graph. Memory networks include recurrent neural networks, Hopfield networks, Boltzmann machines, restricted Boltzmann machines [22], and so on. A memory network can be considered a program with improved computational and memory capabilities. To increase the memory capacity of such networks, external memory units and read/write mechanisms can be integrated to store intermediate network states.

Graph networks The inputs of feedforward and memory networks are typically represented as vectors or sequences of vectors. However, in many practical applications, the data takes the form of graph structures, such as knowledge graphs, social networks, or molecular networks. Handling such graph-structured data can be challenging for feedforward and memory networks. Graph neural networks [110] are neural networks defined on graph-structured data. Each node in a graph consists of a neuron or a group of neurons and the connections between nodes can be directed or undirected. Each node can receive information from neighboring nodes or from itself. Graph networks are a generalization of feed-forward and memory networks and contain many different implementations, such as Graph Convolutional Networks (GCN) [111], Message Passing Neural Networks (MPNN) [112], and many others.

Examples of network structures for feedforward networks, memory networks, and graph networks are given in Figure 2.13, where a circular node represents a neuron, a square node represents a group of neurons, and an arrow represents the flow of information. It should be noted that in real-world applications, most networks are composite structures, i.e., a neural network can contain multiple network structures.

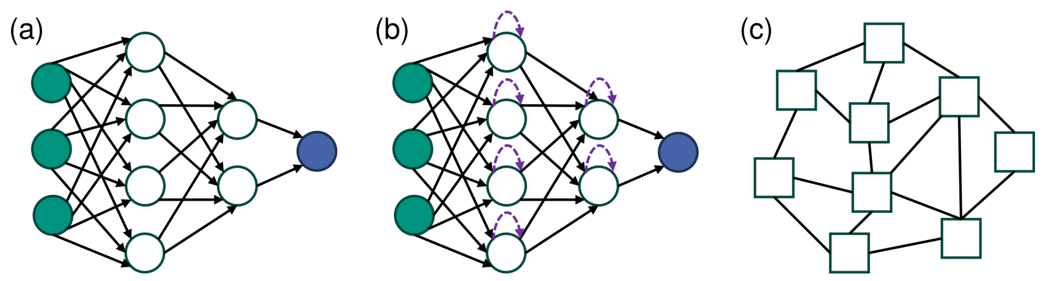


Figure 2.13: Examples of basic network structures for feedforward networks (a), memory networks (b), and graph networks (c).

2.6.1 Feedforward neural network

The artificial neuron, short for neuron, serves as the fundamental component of a neural network, mimicking the structure and features of biological neurons [34]. It accepts a set of input signals and generates outputs. Suppose a neuron receives D inputs $x_1, x_2, \cdots, x_D$, such that the vector $\boldsymbol{x} = [x_1; x_2; \cdots; x_D]$ represents this set of inputs. The net input received by a neuron $z \in \mathbb{R}$ is denoted by the weighted sum of the input signals of $\boldsymbol{x}$:

$$z = \sum_{d=1}^{D} w_d x_d + b = \boldsymbol{w}^\intercal \boldsymbol{x} + b \tag{2.42}$$

where $\boldsymbol{w} = [w_1; w_2; \cdots; w_D] \in \mathbb{R}^D$ is the D-dimensional weight vector and $b \in \mathbb{R}$ is the bias. The net input z yields the neuron's activation value a by a nonlinear function $a = f(z)$, which is called the activation function. Commonly used activation functions are: Sigmoid functions (logistic function or tanh function), Rectified Linear Unit (ReLU) [113, 114] and its variants Leaky ReLU [115], Exponential Linear Unit (ELU) [116], Swish function [117], Gaussian Error Linear Unit (GELU) [118] and so on. An example of a typical neuron structure is given in Figure 2.14.

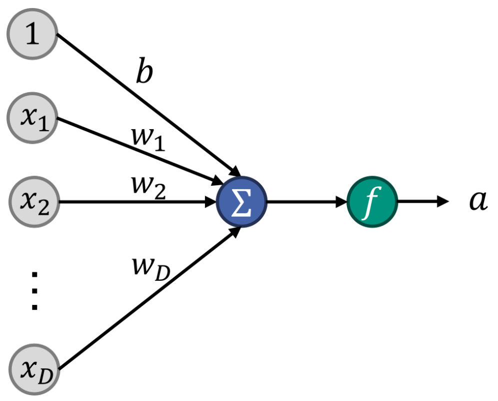


Figure 2.14: Schematic illustration of the neuron structure.

Given a set of neurons, a network can be constructed using the neurons as nodes. Different neural network models have different topologies of network connections. A comparatively simpler topology is the feedforward network. Feedforward Neural Networks (FNN) were the first artificial neural networks invented and are sometimes called Multi-Layer Perceptron (MLP). In a FNN, neurons are spread across multiple layers and neurons in each layer receive signals from neurons in the previous layer and produce signals as output to the next layer. Usually, the first layer is called the input layer, the last layer is called the output layer, and the other intermediate layers are called hidden layers. No feedback is present throughout the network, and the signal flows unidirectionally from the input layer to the output layer, forming a directed acyclic graph.

With input layer $\boldsymbol{a}^{(0)} = \boldsymbol{x}$, the FNN continuously iterates the following equation to propagate information:

$$\begin{aligned} \boldsymbol{z}^{(l)} &= \boldsymbol{W}^{(l)}\boldsymbol{a}^{(l-1)} + b^{(l)}, \\ \boldsymbol{a}^{(l)} &= f_l(\boldsymbol{z}^{(l)}). \end{aligned} \tag{2.43}$$

Firstly, the net activation value $\boldsymbol{z}^{(l)}$ of the neuron in the l-th layer is calculated based on the activation value $\boldsymbol{a}^{(l-1)}$ of the neuron in the $(l-1)$-th layer, and then it is passed through an activation function to get the activation value of the neuron in the l-th layer. $\boldsymbol{W}^{(l)}$ is the weight matrix and $b^{(l)}$ is the bias from $(l-1)$-th layer to l-th layer. Therefore, each neural layer can also be considered as an affine transformation and a nonlinear transformation. In this way, the FNN can pass the information layer by layer to get the final output of the network $\boldsymbol{a}^{(L)}$ and the whole network can be regarded as a composite function $\phi(\boldsymbol{x}; \boldsymbol{W}, \boldsymbol{b})$:

$$\boldsymbol{x} = \boldsymbol{a}^{(0)} \rightarrow \boldsymbol{z}^{(1)} \rightarrow \boldsymbol{a}^{(1)} \rightarrow \boldsymbol{z}^{(2)} \rightarrow \cdots \rightarrow \boldsymbol{a}^{(L-1)} \rightarrow \boldsymbol{z}^{(L)} \rightarrow \boldsymbol{a}^{(L)} = \phi(\boldsymbol{x}; \boldsymbol{W}, \boldsymbol{b}) \tag{2.44}$$

where $\boldsymbol{W}, \boldsymbol{b}$ denotes the connection weights and biases of all layers in the network.

2.6.2 Convolutional neural network

Convolutional Neural Network (CNN or ConvNet) [119, 120] is usually a deep feedforward neural network with local connectivity and weight sharing. CNNs were initially employed for processing image data instead of fully connected feedforward networks which could yield two major issues. Firstly, as the number of neurons in the hidden layer increases, the size of the parameters also dramatically increases. Consequently, the training efficiency of the neural network decreases and is prone to overfitting. Secondly, objects in natural images possess local invariant features, including scale, translation, and rotation, which bear no impact on their semantic information. Nonetheless, fully connected feedforward networks struggle to extract these local invariant features, necessitating data enhancement to improve performance.

CNNs are based on the biological mechanism of the receptive field [121]. This mechanism refers to certain neurons in the visual (or auditory) nervous systems that only receive signals from the region they innervate. A neuron's receptive field is a specific area of the retina, and only stimuli within this area can activate the neuron. Currently, CNNs are usually feedforward neural networks consisting of convolutional, pooling, and fully connected layers stacked on top of each other. CNNs possess three structural properties, namely local connectivity, weight sharing, and gathering, enabling them to exhibit certain degrees of translation, scaling, and rotation invariance [120]. CNNs are mainly used for various tasks in image and video analysis (e.g., image classification, face recognition, object recognition, image segmentation), and their accuracies are generally higher than those of other neural network architectures [34]. An overview of some commonly used CNN architectures is shown in Figure 2.15.

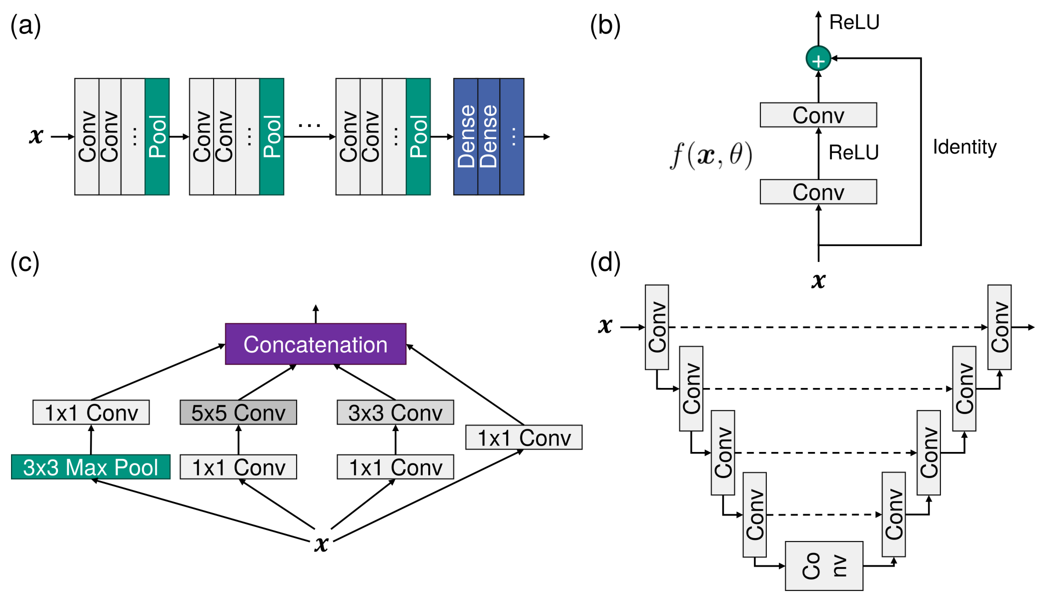


Figure 2.15: Overview of the most commonly used CNN architectures: (a) AlexNet/VGGNet, (b) ResNet, (c) Inception-Net, (d) U-Net.

AlexNet [24] is the first modern deep convolutional network model, which uses many modern deep convolutional network techniques for the first time, such as the use of Graphics Processing Units (GPUs) for parallel training, the adoption of ReLU as a nonlinear activation function, the use of dropout layer to reduce overfitting, and the use of data augmentation to improve the accuracy of the model. VGGNet [122] is an extension to AlexNet and it was developed to increase the depth of such CNNs in order to increase the model performance. In this architecture, a convolutional network can be stacked with some consecutive convolutional blocks (convolutional layers and pooling layers) followed by some fully connected layers. A schematic illustration of these structures is shown in Figure 2.15a.

Residual Network (ResNet) [123] adds a shortcut connection (also known as a residual connection) to a nonlinear convolutional layer to improve the efficiency of information propagation. Suppose that in a deep network, a nonlinear unit $f(\boldsymbol{x};\boldsymbol{\theta})$ with parameters $\boldsymbol{\theta}$ (which can be one or many layers of convolutional layers) is expected to approximate an objective function of $h(\boldsymbol{x})$ and the objective function can be split into two parts: the identity function $\boldsymbol{x}$ and the residue function $h(\boldsymbol{x}) - \boldsymbol{x}$.

$$h(\boldsymbol{x}) = \underbrace{\boldsymbol{x}}_{\text{Identity Function}} + \underbrace{\left(h(\boldsymbol{x}) - \boldsymbol{x}\right)}_{\text{Residue Function}} . \tag{2.45}$$

According to the universal approximation theorem [108, 109], a nonlinear unit consisting of a neural network has sufficient ability to approximate either the original objective function or the residue function, but in practice, the latter is easier to learn [123]. Therefore, the original optimization problem can be converted into: having the nonlinear cell $f(\boldsymbol{x};\boldsymbol{\theta})$ to approximate the residual function $h(\boldsymbol{x}) - \boldsymbol{x}$, and using $f(\boldsymbol{x};\boldsymbol{\theta}) + \boldsymbol{x}$ to approximate $h(\boldsymbol{x})$.

A typical example of a residual unit is given in Figure 2.15b. The residual unit consists of multiple cascaded (equal-width) convolutional layers and a directly connected edge across the layers, which is then activated by, e.g., a ReLU function to obtain the output.

Inception network is designed to solve the critical issue of setting an appropriate kernel size of a convolutional layer by using a block that contains multiple convolutional operations with different sizes, which is called inception module [124]. An inception network consists of multiple inception modules and a small number of convergence layers stacked together. Inception modules simultaneously use convolution kernels with different sizes, such as 1×1, 3×3, 5×5, and the resulting feature maps are stacked together in-depth as the output feature maps (shown in Figure 2.15c).

U-Net can achieve very good performance in different segmentation tasks, e.g. for biomedical applications [125]. It requires only a small number of labeled images and is highly efficient with low training time. A typical U-Net architecture is shown in Figure 2.15d and it has an encoder-decoder structure, where the left half of the encoder consists of a number of convolutions, pooling, and down-sampling of the image, and the decoder on the right side performs the up-sampling and restores the shape of the original image, giving a prediction for each pixel. Compared to the traditional CNN architecture, U-Net utilizes skip connections at the same stage rather than directly supervising high-level semantic features. This guarantees that the final recovered feature maps comprise a higher quantity of low-level features and permits the fusion of features from different scales for multi-scale prediction. Moreover, using multiple up-sampling techniques improves the granularity of the segmentation map, allowing for more precise recovery of edges and other important information [126].

2.6.3 Recurrent neural network

Feedforward neural networks transfer information in one direction, which limits their ability to some extent despite their ease of learning. In contrast, biological neural networks feature more complex inter-neuron connections. Essentially, a feedforward neural network works as a complex mathematical function, with each individual input determining the corresponding output. In real-world applications, network output is often determined not only by the current input but also by its past output. Additionally, it is challenging for feedforward networks to deal with temporal (time-series) or sequential data, such as videos, speeches, and texts. The length of time-series data is typically short and variable. However, feedforward neural networks necessitate fixed dimensions for both inputs and outputs and do not allow for arbitrary changes. Therefore, to process time-series data, it is necessary to use more capable models with different connectivity architectures.

Recurrent Neural Networks (RNNs) [127] are neural network models with short-term memory, where neurons receive input from other neurons as well as themselves, creating a network structure with loops. RNNs have been widely used in speech recognition and natural language generation. The parameters of RNNs can be learned using the backpropagation over time algorithm [128]. An example of an RNN is given in Figure 2.16.By using neurons with self-feedback, it is expected to be able to process time-series data of arbitrary length. Given an input sequence $\boldsymbol{x}_{1:T} = (\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_t, \cdots, \boldsymbol{x}_T)$, RNN updates the activity value $\boldsymbol{h}_t$ of the hidden layer with feedbacks by the following:

$$\boldsymbol{h}_t = f(\boldsymbol{h}_{t-1}, \boldsymbol{x}_t) \tag{2.46}$$

where $\boldsymbol{h}_0 = 0$ and $f(\cdot)$ is a nonlinear function that can be a feedforward network.

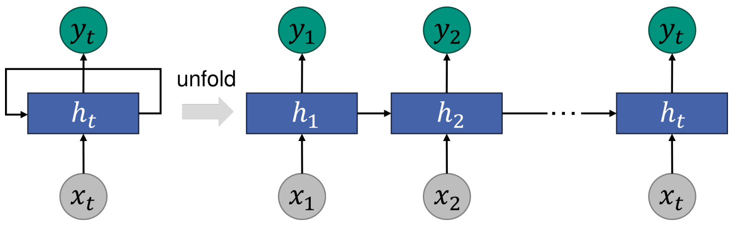


Figure 2.16: Schematic illustration of RNN.

Mathematically, Equation 2.46 represents a dynamical system wherein the state of the system undergoes changes over time following a certain pattern. In simple terms, a dynamical system describes how all points in a given space (for instance, the state space of a physical system) change with time. Many real-world phenomena such as the swing of a pendulum or the trajectory of a billiard ball can be elucidated using dynamical systems. In theory, RNNs can approximate any nonlinear dynamical system.

Long-term dependency problem

RNNs face a significant challenge during the learning process, which is capturing long-term state dependencies, as they are more vulnerable to the gradient vanishing or explosion problem, also known as the long-term dependency problem [129, 130].

In order to address this issue, a potential solution is to implement a gating mechanism that regulates the rate at which information is accumulated. This can involve selectively adding new information and selectively forgetting previously obtained information, to ensure optimal control and management of the information flow. Such a type of network can be called a Gated Recurrent Neural Network (Gated RNN). The most popular two types of Gated RNN are Long Short-Term Memory (LSTM) networks [130] and Gated Recurrent Unit (GRU) networks [131, 132]. LSTMs contain a number of "cells" in the hidden layer of the neural network, with three gates: an input gate, an output gate, and a forgetting gate. These gates control the flow of information needed to predict the output of the network. GRU is a variant of RNN and is similar to LSTM in that it is also designed to solve the short-term memory problem. However, instead of having three gates, it uses only two: a reset gate and an update gate. Similar to gates in LSTM, the reset and update gates control what and how much information is retained.

Parallelization problem

In addition to the long-term dependency problem, another drawback of RNNs is that they cannot be computed in parallel. To enhance parallel computation efficiency and improve long-term dependency capture, the transformer model [133], a self-attention-based machine learning model for sequence-to-sequence processing, has been introduced and gained significant popularity. The attention mechanism finds many applications in different areas of deep learning, including Computer Vision (CV) for sensory field capture on images and Natural Language Processing (NLP) for feature or token localization. The transformer model solely comprises of the attention component, free of any CNN or RNN layers.

The attention mechanism is implemented to account for the fact that the computation of RNN is constrained to being sequential, which brings two problems: (i) the computation of time slice t relies on the results of the computation at time $t-1$, which limits the parallelism of the model; (ii) information can be lost during sequential computation, although gating mechanisms such as LSTM mitigates the problem of long-term dependency to some extent, they do not perform well for particularly long-term dependency situations. The transformer model can resolve these two issues. Firstly, it utilizes the attention mechanism to decrease the distance between any two positions in a sequence to a constant. Secondly, it is not a sequential structure resembling RNN, leading to enhanced parallelism. The overall structure of the transformer model comprises multiple repetitions of both the encoder and decoder components, which can facilitate the model's ability to deal with large amounts of sequential data such as texts.

In the encoder of the transformer, the input data (usually a vector sequence with words $\boldsymbol{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_T]$) will first go through a module called "self-attention" to get a weighted feature $\boldsymbol{Z}$:

$$\text{self-attention}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \boldsymbol{V}\text{softmax}\Big(\frac{\boldsymbol{K}^\intercal \boldsymbol{Q}}{\sqrt{D_k}}\Big) \tag{2.47}$$

where $\boldsymbol{Q}$ is the query matrix, $\boldsymbol{K}$ is the key matrix, $\boldsymbol{V}$ is the value matrix, and D_k is the dimension of $\boldsymbol{Q}$ and $\boldsymbol{K}$. The softmax function is used to normalize the output to a probability distribution [22]. These three matrices are obtained by multiplying the inputs $\boldsymbol{X}$ by three different weight matrices $\boldsymbol{W}^Q$, $\boldsymbol{W}^K$, and $\boldsymbol{W}^V$. The concept of query, key, and value is taken from the information retrieval system. Using an online search as an example: when searching for a product on an e-commerce platform, the user inputs a query into the search engine. The search engine then matches the query with key information such as product type, color, and description and retrieves the matching value based on the similarity between the query and the key. The self-attention $(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V})$ plays a similar role to this concept [134].

The embedded $\boldsymbol{Z}$ obtained after the attention operation is sent to the next module of the encoder, i.e., Feedforward Neural Network (FNN). This fully connected has two layers, the first with a ReLU activation function and the second with a linear activation function. For a vector at each position in the input sequence $\boldsymbol{z} \in \boldsymbol{Z}$:

$$\text{FNN}(\boldsymbol{z}) = \text{ReLU}(\boldsymbol{z}\boldsymbol{W}_1 + \boldsymbol{b}_1)\boldsymbol{W}_2 + \boldsymbol{b}_2 \tag{2.48}$$

where $\boldsymbol{W}$ and $\boldsymbol{b}$ are parameters of the model.

The decoder differs from the encoder in that it has an additional "encoder-decoder attention" module. The two attentions are used to compute the weights of the inputs and outputs, respectively. The self-attention is used to learn the relationship between the current word and the previously outputted words. In self-attention, the query, keys, and values come from the output of the previous decoder block. On the other hand, the encoder-decoder attention is used to learn the relationship between the current word and the encoded feature vector. The query for the encoder-decoder attention comes from the output of the previous decoder block, while the keys and values come from the output of the whole encoder.

The transformer model introduced thus far lacks the ability to capture sequential sequences. This means that regardless of how much the sentence structure is disrupted, the transformer will produce similar results because self-attention abandons sequential operations in favor of parallel computation. To address this issue, the original paper introduces the feature of position embedding, which injects absolute or relative positional information when encoding word vectors so that the model can distinguish words in different positions [135]. The transformer's design achieves its biggest performance improvement by setting the distance equivalence of any two words to be the same. This approach is highly effective in solving the tricky long-term dependency problem. An illustrative structure of the transformer is shown in Figure 2.17.

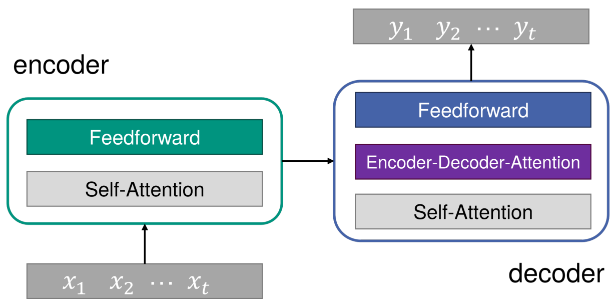

y_1 y_2 $\cdots$ y_t
encoder
Feedforward
Self-Attention
Feedforward
Encoder-Decoder-Attention
Self-Attention
x_1 x_2 $\cdots$ x_t
decoder

Figure 2.17: Schematic illustration of the transformer architecture.

2.6.4 Graph neural networks

In practical applications, many data are graph-structured. Dealing with such data is challenging for both feedforward and feedback networks. To address these difficulties, a type of neural network known as Graph Neural Networks (GNNs) [136, 110] has been developed. GNNs extend the concept of message passing to graph-structured data, providing a solution for analyzing graph-structured data. The primary focus of GNNs is the graph structure, which can be observed extensively in daily life and scientific research, including interpersonal relationships, social networks, web pages, research paper citations, and molecular structure in chemistry. Consequently, these networks have a wide range of application scenarios.

Since the initial proposal of GNN, various model architectures have been developed. These models can be classified into different categories including recurrent graph neural networks, Graph Convolution Networks (GCN), graph attention networks, graph autoencoders, and graph spatial-temporal networks [110, 136, 137]. They have made a lot of achievements in a variety of fields such as recommendation systems, knowledge graphs, images, and texts. Although the architectures of these GNNs are different, each has its own advantages and disadvantages, the core objective is the same, that is, how to extract features from graph structures.

An arbitrary graph structure can be represented as $G(\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ denotes the set of nodes, $\mathcal{E}$ denotes the set of edges, and each edge denotes a connection between two nodes, which can be directed or undirected. Each node v in the graph uses a set of neurons to represent its state $\boldsymbol{h}^{(v)}$, and the initial state can be the input feature $\boldsymbol{x}^{(v)}$ of node v. Each node can update its own state at t-th moment by receiving messages from itself and neighboring nodes at $(t-1)$-th moment (past state), that is:

$$\begin{aligned} \boldsymbol{m}_t^{(v)} &= \sum_{u \in \mathscr{N}(v)} f(\boldsymbol{h}_{t-1}^{(v)}, \boldsymbol{h}_{t-1}^{(u)}, \boldsymbol{e}^{(u,v)}), \\ \boldsymbol{h}_t^{(v)} &= g(\boldsymbol{h}_{t-1}^{(v)}, \boldsymbol{m}_t^{(v)}), \end{aligned} \tag{2.49}$$

where $\mathscr{N}(v)$ denotes the neighbors of the node v, $\boldsymbol{m}_t^{(v)}$ denotes the messages (information) received by the node v at the t-th moment, $\boldsymbol{e}^{(u,v)}$ is the feature on the edge $e^{(u,v)}$, $f(\cdot)$ and $g(\cdot)$ are usually non-linear functions, in this case, neural network models.

The formula above represents a technique for synchronously updating the graph, where all units receive information and simultaneously update their state. For directed graphs, using RNNs is usually a more effective asynchronous method for updating the graph. After the entire graph has been updated T times, a representation of the entire network can be obtained by a readout function $\mathrm{g}(\cdot)$:

$$\boldsymbol{o}_t = \mathrm{g}\Big(\{\boldsymbol{h}_T^{(v)} \mid v \in \mathcal{V}\}\Big) \tag{2.50}$$

and a schematic diagram of this process is shown in Figure 2.18.

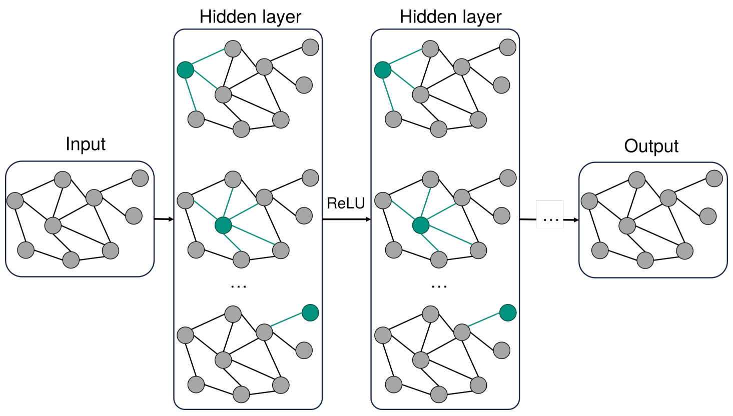


Figure 2.18: Schematic illustration of GNN.

2.6.5 Generative models

Deep generative models are a category of machine learning models that use deep neural networks to produce data, frequently in the form of images, text, or complex structures. Their objective is to capture and reproduce the underlying patterns and structures of the data they are trained on, allowing for tasks like image generation, text synthesis, and beyond. The most popular examples include Variational AutoEncoders (VAEs) [138, 139] and Generative Adversarial Networks (GANs) [140], which have revolutionized the fields of computer vision, natural language processing, and creative content generation. Furthermore, in recent years, other generative models like the flow-based model [141], diffusion model [142], and Generative Pre-trained Transformers (GPTs) [143] have also become more and more popular in generation tasks.

Variational AutoEncoders (VAEs) are a family of generative neural network models that can learn complex probabilistic distributions of data characterized by latent variables [138, 139]. They are usually seen as an extension to the AutoEncoders (AEs) models for their architectural similarity, though VAEs actually belong to the families of probabilistic graphical models and variational Bayesian methods whose goal and mathematical formulation are significantly different from the standard AEs. Both the VAEs and AEs are composed of two sub-models: an encoder and a decoder. The encoder $\boldsymbol{z} = f(\boldsymbol{x})$ compresses (usually high-dimensional) input data $\boldsymbol{x}$ into a low-dimensional multivariate latent variable $\boldsymbol{z}$ while the decoder $\boldsymbol{x}' = g(\boldsymbol{z})$ reconstructs the latent representation $\boldsymbol{z}$ back to the original data space to generate the output data $\boldsymbol{x}'$. AEs are trained to minimize the discrepancy between inputs $\boldsymbol{x}$ and their corresponding reconstructions $\boldsymbol{x}'$, however, it has the problem of being prone to generate new outputs with unrealistic or invalid topology.

The basic VAE was introduced to solve this problem by restricting the distribution of the encoded latent variables $\boldsymbol{z}$ from the input data to the distribution that is used for generating the output. This indirectly enforces consistency between the model outputs and the input data. The model can generate new samples that have similar characteristics as those in the original data set by sampling the latent variables from the learned distribution. That is why this model is frequently referred to as a generative model.

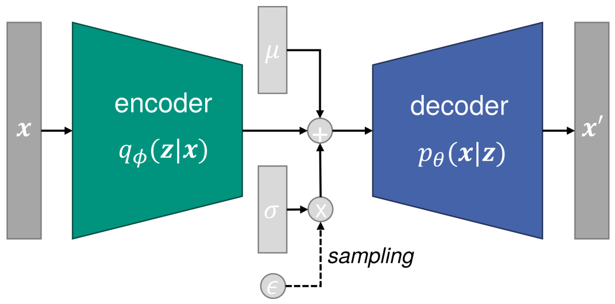
Let the input data set be $\mathcal{D} = \{\boldsymbol{x}^{(n)}\}_{n=1}^{N}$, which is characterized by true data distribution $p(\boldsymbol{x})$ and it is the objective to model. In practical applications, data can be exceedingly complex, and its mathematical expression is frequently unknown. This can pose significant challenges to accurately model the distribution of such data. In probabilistic graphical models, the observed data $\boldsymbol{x}$ is assumed to be generated from some latent variables $\boldsymbol{z}$. This generation process can be divided into two steps: first, latent variables are sampled from their prior distribution $p_{\boldsymbol{\theta}}(\boldsymbol{z})$, and second, data is generated from these sampled latent variables $p_{\boldsymbol{\theta}}(\boldsymbol{x} \mid \boldsymbol{z})$ with $\boldsymbol{\theta}$ as parameters. $p_{\boldsymbol{\theta}}(\boldsymbol{z})$ and $p_{\boldsymbol{\theta}}(\boldsymbol{x} \mid \boldsymbol{z})$ are usually assumed to be some parameterized family of distributions, such as Gaussian distribution. Hence, matching the model distribution with the data distribution is equivalent to maximizing the marginal likelihood $p_{\boldsymbol{\theta}}(\boldsymbol{x})$:

$$p_{\boldsymbol{\theta}}(\boldsymbol{x}) = \int_{\boldsymbol{z}} p_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{z}) d\boldsymbol{z} = \int_{\boldsymbol{z}} p_{\boldsymbol{\theta}}(\boldsymbol{x} \mid \boldsymbol{z}) p_{\boldsymbol{\theta}}(\boldsymbol{z}) d\boldsymbol{z} = \mathbb{E}_{\boldsymbol{z} \sim p_{\boldsymbol{\theta}}(\boldsymbol{z})} \left[p_{\boldsymbol{\theta}}(\boldsymbol{x} \mid \boldsymbol{z}) \right]. \tag{2.51}$$

Given an observed data sample $\boldsymbol{x}$, its latent variable $\boldsymbol{z}$ can be inferred from its posterior $p_{\boldsymbol{\theta}}(\boldsymbol{z} \mid \boldsymbol{x})$. However, this distribution is usually hard to compute and can even be intractable. To make it feasible and speed up the calculus, in variational inference, another distribution $q_{\boldsymbol{\phi}}(\boldsymbol{z} \mid \boldsymbol{x})$ with parameters $\boldsymbol{\phi}$ from a known family is introduced to approximate the posterior distribution $p_{\boldsymbol{\theta}}(\boldsymbol{z} \mid \boldsymbol{x})$ by minimizing the KL divergence [37] between these two distributions (generally assumed to be Gaussian): $D_{KL}\left(q_{\boldsymbol{\phi}}(\boldsymbol{z} \mid \boldsymbol{x}) \mid\mid p_{\boldsymbol{\theta}}(\boldsymbol{z} \mid \boldsymbol{x})\right)$. In this way, the overall problem is translated into the autoencoder domain, in which the conditional likelihood distribution $p_{\boldsymbol{\theta}}(\boldsymbol{x} \mid \boldsymbol{z})$ is carried by a decoder neural network model, while the approximated posterior distribution $q_{\boldsymbol{\phi}}(\boldsymbol{z} \mid \boldsymbol{x})$ is computed by another encoder neural network model. However, the computation of the KL divergence still involves the same problem of intractable integrals as in the posterior computation. To solve this, the original formula is deformed to derive an equivalent function to optimize, which is usually referred to as the Evidence Lower BOund (ELBO) and is written as:

$$\begin{aligned} L_{\boldsymbol{\theta},\boldsymbol{\phi}} &= \log p_{\boldsymbol{\theta}}(\boldsymbol{x}) - D_{KL}\left(q_{\boldsymbol{\phi}}(\boldsymbol{z} \mid \boldsymbol{x}) \parallel p_{\boldsymbol{\theta}}(\boldsymbol{z} \mid \boldsymbol{x})\right) \\ &= \mathbb{E}_{\boldsymbol{z} \sim q_{\boldsymbol{\phi}}(\boldsymbol{z} \mid \boldsymbol{x})} \left[\log p_{\boldsymbol{\theta}}(\boldsymbol{x} \mid \boldsymbol{z})\right] - D_{KL}\left(q_{\boldsymbol{\phi}}(\boldsymbol{z} \mid \boldsymbol{x}) \parallel p_{\boldsymbol{\theta}}(\boldsymbol{z})\right). \end{aligned} \tag{2.52}$$

Maximizing ELBO is equivalent to maximizing the log-likelihood of the observed data and minimizing the KL divergence at the same time. The above function can also be interpreted as two parts: the first term is usually called the reconstruction loss between the input and the output of the network, which is the expected log-likelihood of reconstructing the same input $\boldsymbol{x}$ flowing through the model; the second term can be seen as a regularization posed on the posteriors, which will penalize when it deviates from the prior. To solve the optimization problem of equation (2.52) using the universal gradient descent approach by backpropagation, a reparameterization trick is needed to remove the stochastic sampling from the formation and thus make the training process differentiable. A schematic diagram of this model is shown in Figure 2.19.



$\boldsymbol{x}$
encoder
$q_\phi(\boldsymbol{z}|\boldsymbol{x})$
μ
σ
+
×
sampling
ϵ
decoder
$p_\theta(\boldsymbol{x}|\boldsymbol{z})$
$\boldsymbol{x}'$

Figure 2.19: Schematic illustration of VAE.

There are many variants to the original VAE model, which have been used to adapt the architecture to other domains and improve its performance. β-VAE [144, 145] is an implementation with a weighted KL-divergence term to better discover disentangled latent variables and interpret these representations. The Conditional VAE (CVAE) [146] can add label information into the model (transforming the task into a supervised one), forcing a deterministic constrained representation of the learned data. Some variants change the structure of latent space to further improve the representation learning, such as VQ-VAE [147]. There are also some architectures that mix VAEs and Generative Adversarial Networks (GANs) to obtain hybrid models [148].

Generative Adversarial Networks (GANs) are a type of generative models developed by Goodfellow et al. [140] which learns to implicitly represent the Probability Distribution Function (PDF) of a given data set (i.e., p_{data}). Since p_{data} is unknown, the result of the learning process is an estimate of the PDF called p_{model} from which a set of samples can be generated, that is, they learn a function that can sample from p_{model}, which reasonably approximates those from the real data set (p_{data}). The idea of GANs originated from game theory, as a two-player game between the generator and the discriminator.

Specifically, GAN trains a generator network $G(\boldsymbol{z})$ that produces samples $\boldsymbol{x}_G$ from latent variables $\boldsymbol{z}$ to approximate real samples $\boldsymbol{x}_{data}$, and a discriminator network $D(\boldsymbol{x})$ that distinguishes the generated samples from the real samples. GAN will identify the latent variables of data by training a generator-discriminator model pair in an adversarial manner. A more vivid analogy of GAN is given by its original author [149] that in the adversary scenario, the generator can be thought of as a group of counterfeiters who try to produce fake currency, while the discriminator is analogous to a team of police, trying to detect the counterfeit currency from the real money. Competitions in this adversary game would keep pushing both sides to the equilibrium in which the counterfeits are indistinguishable. When the generator is capable of producing realistic samples at the equilibrium, the latent variable $\boldsymbol{z}$ would be naturally taken as the “code” of the data. A schematic diagram of this model is shown in Figure 2.20.

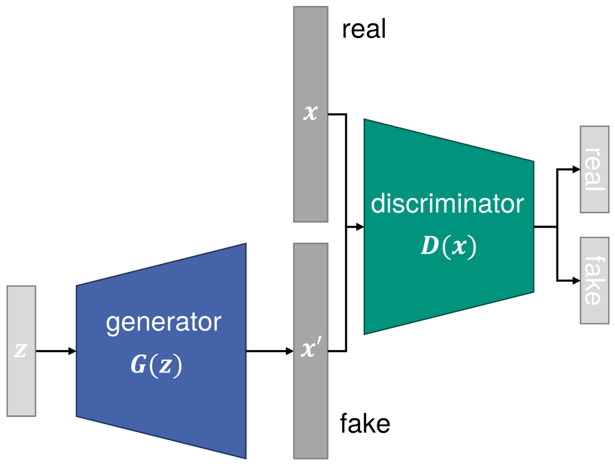


Figure 2.20: Schematic illustration of GAN.

The training process of GAN consists of a min-max game between two functions, the generator $G(\boldsymbol{z})$ and the discriminator $D(\boldsymbol{x})$ [140]. $G(\boldsymbol{z})$ maps a D-dimensional latent vector $\boldsymbol{z} \in \mathbb{R}^D$ from noise distribution $p(\boldsymbol{z})$ (typically a Gaussian or a uniform distribution) to a point in the space of real data with parameters $\boldsymbol{\theta}^{(G)}$ as $G(\boldsymbol{z}; \boldsymbol{\theta}^{(G)})$, while $D(\boldsymbol{x})$ represents the probability that $\boldsymbol{x}$ comes from p_{data}. The aim of the training is to make the implicit density learned by $G(\boldsymbol{z})$ (i.e., p_{model}) to be close to the distribution of real data (i.e., p_{data}). Through this adversarial game, the two players are in constant competition, where the discriminator attempts to label the real and fake data correctly, i.e., $D(\boldsymbol{x}_{data}) = 1$ and $D\big(G(\boldsymbol{z})\big) = 0$, while the generator tries to fool the discriminator into misclassifying synthetic data as real, i.e. $D\big(G(\boldsymbol{z})\big) = 1$. This problem is formally defined as a min-max game between the generator and discriminator where the objective function to optimize is $V(G, D)$:

$$\min_G \max_D V(G, D) = \mathbb{E}_{\boldsymbol{x} \sim p_{data}(\boldsymbol{x})}\Big[\log D(\boldsymbol{x})\Big] + \mathbb{E}_{\boldsymbol{z} \sim p(\boldsymbol{z})}\Big[\log\Big(1 - D\big(G(\boldsymbol{z})\big)\Big)\Big] \tag{2.53}$$

This competitive game would eventually lead to a Nash equilibrium [150] between the generator G and the discriminator D. Throughout the learning process, the generator learns to represent the probability distribution function p_{model} which is as close as possible to the distribution of the real data $p_{data}(\boldsymbol{x})$. At the Nash equilibrium, the samples of $\boldsymbol{x} = G(\boldsymbol{z}) \sim p_{model}(\boldsymbol{z})$ are indistinguishable from the real samples $\boldsymbol{x} \sim p_{data}(\boldsymbol{x})$, thus $p_{model}(\boldsymbol{z}) = p_{data}(\boldsymbol{x})$ and $D(\boldsymbol{x}) = \frac{1}{2}$ for all $\boldsymbol{x}$ since the discriminator can no longer distinguish between real and synthetic data.

In theory, a Nash equilibrium is achieved after sufficient training, however, in practice, this is not always the case. GANs have shown instability during training that can lead to mode collapse [149, 151]. One important improvement to help train the model was proposed in the work [152], known as Wasserstein GAN (WGAN). According to their research, the training difficulty arises from the divergence being non-continuous with respect to the generator's parameters. To overcome this, they suggest applying the earth mover's distance (also referred to as the Wasserstein distance) to measure the difference between probability distributions, which is continuous under reasonable assumptions and suits better in situations when two probability distributions have no overlaps. This change can lead to a more stable convergence of GAN, resulting in better results.

A direct extension of the GAN is Deep Convolutional Generative Adversarial Networks (DCGAN) [153], except that it explicitly uses convolutional and transposed convolutional layers instead of the multilayer perceptron in the original GAN in the discriminator and generator, respectively.

In the original GAN, there is no control over what to generate as the output depends only on the latent variable of random noise. To improve this, a conditional input $\boldsymbol{c}$ can be added to the random noise $\boldsymbol{z}$ so that the generated image is defined by $G(\boldsymbol{c}, \boldsymbol{z})$. This kind of neural network is named conditional GAN (cGAN)[154]. Usually, the conditional input vector $\boldsymbol{c}$ is directly concatenated to the noise vector $\boldsymbol{z}$, and the new resultant vector is used as the input to the generator. Furthermore, an auxiliary output model can be added into the architecture to enhance the model's capabilities, such as Auxiliary Classifier GAN (ACGAN) [155]. On the other hand, there are also unsupervised ways to improve the latent space, such as InfoGAN [156], which works by decomposing the latent vector into an incompressible noise part and an informative label part, and encourages maximizing mutual information.

In addition to the above, variants of GAN (e.g., pix2pix [157] and cycleGAN [158]) are also usually used to perform translations between two domains, such as image-to-image translation or style transfer.

Chapter 3

Methodology and workflow for machine learning in materials science

This chapter presents a unified methodology and workflow for effectively applying Machine Learning (ML) in materials research, which provides a foundational understanding of these ideas by introducing basic concepts and steps supported by an extensive review of diverse state-of-the-art application examples.

3.1 Generic principle and methodology

ML is the branch of Artificial Intelligence (AI) concerned with the development of algorithms and models that can automatically learn patterns from data and perform tasks without explicit instructions or rules [159]. While ML models and algorithms have been known since the 1950s, it is only in the last decade that the exponential increase in the amount of data, coupled with the great improvement in computing power (especially GPUs), make modern ML methods achieve state-of-the-art performance in the mainframe of computer science, and therefore they have attracted wider attention and interest than ever before. They have also been adopted and extended to many other fields, such as physics and chemistry, and have achieved good results and are breaking many new frontiers. ML is also increasingly used to assist researchers in exploring material relationships, predicting novel materials or properties from existing data, and understanding material chemistry, due to its ability to solve complex tasks in a largely autonomous manner [17].

The fundamental principle of applying ML models in materials research is to predict the properties of the material system being studied efficiently and accurately, ultimately replacing costly experiments/simulations and minimizing the repetitive, time-consuming work in the analysis of characterization data. In materials research, many ML studies aim to construct a model from available data on the studied material system. This model can efficiently map the relationship between material properties. Once developed, the model allows for quick prediction of the interested aspects of materials (e.g., compositions, structures, and properties), and can be later applied to large data sets to expeditiously screen the most promising candidate for new material discovery, to optimize current ones, or to efficiently guide the search process. These predicted candidates can be validated by standard processes in materials research, which usually involves performing experiments to synthesize samples followed by characterization and testing their performance. In addition, theoretical simulations are often performed to better investigate the underlying mechanisms. The obtained new results can also be fed back into the existing database for further analysis or to improve the performance of ML models, allowing for iterative development of materials until a desired candidate is found. Based on these concepts and ideas, this work proposes a unified methodology and workflow for applying ML in materials research, and a schematic illustration is shown in Figure 3.1.

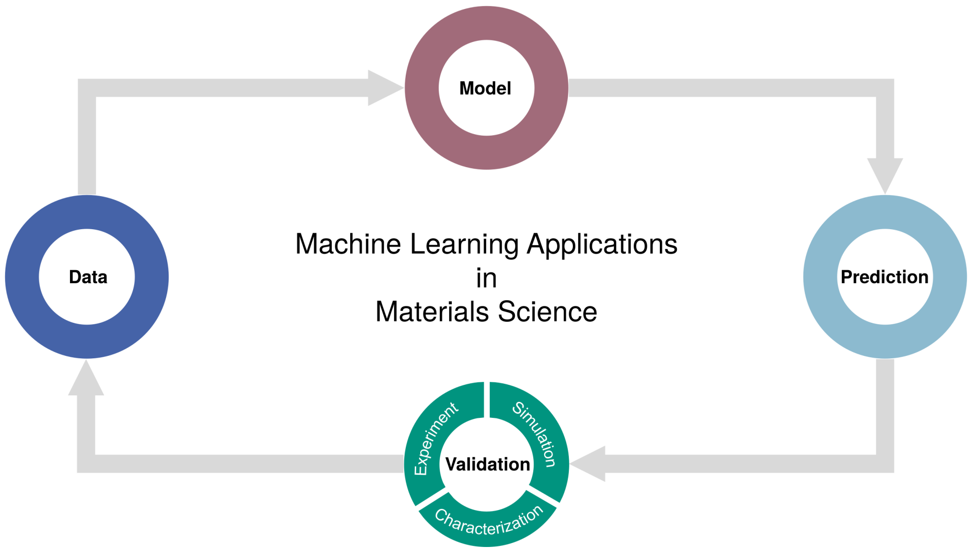


Figure 3.1: Conceptual workflow of applying machine learning in materials science.

3.1.1 Basic workflow

The generic workflow for applying ML techniques in the field of materials science is illustrated in Figure 3.2. This process involves setting a task, preparing data, training and selecting a model, evaluating performance, and applying the model to new data sets.

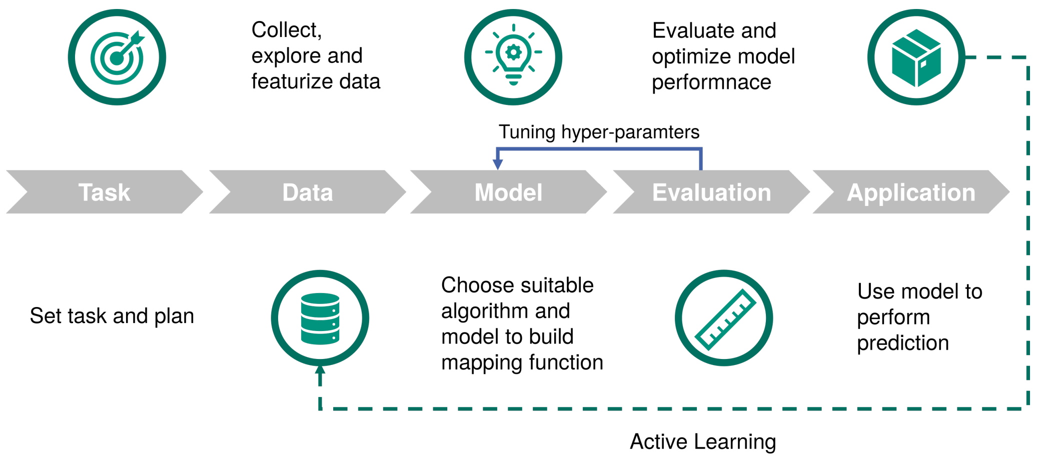

Collect, explore and featurize data
Evaluate and optimize model performnace
Tuning hyper-paramters
Task
Data
Model
Evaluation
Application
Set task and plan
Choose suitable algorithm and model to build mapping function
Use model to perform prediction
Active Learning

Figure 3.2: Workflow for building ML models for material science. The initial step is to identify the task by setting a clear goal for prediction. Next, collect data related to the studied material system and process it in a way that benefits training the ML model. Evaluate the model and select the optimal one, and apply it to fast predictions. In addition, the obtained new results can be added to the existing database for further analysis or to enhance the performance of the ML model for predicting the next promising candidate (referred to as "active learning").

(i) Task identification

The first step in applying ML to materials science involves determining the task by setting the target to predict, which commonly depends on the interests of researchers and the goals of the project. To construct a feasible machine learning model, the selection of objectives should be potentially learnable from the relevant available data, and precisely and unambiguously defined. In materials science, common topics of interest include crystal/molecular structure, composition, elemental information, experimental conditions, other experimentally measured quantities, and characterization data such as microscopy images.

An inappropriate selection of the prediction target may lead to imprecise results, large errors, or limited generation capabilities when applied to new samples, restricting the model's scope and potentially causing faulty behavior. Furthermore, modeling difficulties may arise due to uncertain or inaccurate data from both experimental and computational sources, which in turn can negatively impact the model's performance. The selection of an appropriate model for materials science research often depends on the target to predict and characteristics of the material system being studied. A common objective in this field is to solve classification or regression tasks. A classification task seeks to learn the mapping between inputs and discrete (categorical) targets. Many material properties are considered to be categorical, such as metal or non-metal, superconductor or non-superconductor, and types of crystal systems like cubic, tetragonal, and orthorhombic. In contrast, the regression task aims at learning the mapping between inputs and continuous numerical targets, such as properties of materials like conductivity, formation energies, band gaps, shear modulus, and many others.

(ii) Data preparation

After identifying the target, the next step is to collect and organize the data set about the targeted material system. The quality and quantity of data play an important role in ML applications within materials science [32, 160]. The data set usually contains variables related to material samples that correspond to the desired properties of the materials. These variables are typically known as features, descriptors, or fingerprints and encompass representations of information such as chemical compositions, atomic or molecular properties, and material preparation experimental conditions. Domain expertise is necessary for researchers to pinpoint optimal features by taking into account the characteristics inherent in the materials system under examination.

The data set required for this task typically originates from two primary sources: experiments or computations. Researchers may create data through large-scale and high-throughput experiments or computations, or acquire it from publicly accessible platforms. The data must be of sufficient quantity and quality to comprehensively cover all relevant information pertaining to the materials being studied. The necessary quantity of data typically varies depending on the chosen ML model according to the task. Nonetheless, a common guideline is that more data generally leads to better performance, with a general rule of thumb that more than 50 data points are required for an ML model to function reasonably well [161]. Certain ML models, such as neural networks, may require a much greater quantity of data.

In terms of data quality, a high-quality data set should encompass a wide range of materials and their corresponding physical/chemical properties, preferably with uncertainties associated with the data. For instance, a model for predicting material properties that is trained on data from a limited number of chemical elements is unlikely to perform effectively across the entire periodic table. Likewise, a model that trains on samples synthesized under low-temperature conditions is not expected to provide accurate predictions for samples obtained at high temperatures in the sintering process. Generally, it is important that the collected data should be representative and cover the nature of the studied materials. In addition, it should be noted that data uncertainty should be taken into careful consideration as it has a strong impact on results, which can arise from multiple sources, such as experimental errors, measuring errors, or computational errors resulting from oversimplified approximations or numerical errors.

Over the past few decades, researchers have developed numerous materials databases containing experimental data. Some notable examples include the Pauling file database [162] and the Inorganic Crystal Structure Database (ICSD) [163], which contain crystal/molecular structures. Furthermore, many other databases such as Pearson crystal data [164], the Cambridge structural database [165], Crystal Open Database (COD) [166], [167], ZINC database [168], GDB databases [169], PubChem [170], provide precise and accurate data on various substances and molecules, allowing researchers to explore their properties and potential applications. In addition to the crystallographic databases, numerous other collections of materials data with measured thermodynamic properties and other physical/chemical properties have also been established [171, 172]. Some more comprehensive overviews of materials databases can be found in the reviews [32, 173, 174].

A major challenge in constructing high-quality ML models is the limited size and heterogeneity of existing experimental data repositories in the field of materials science. Inconsistencies in experimental conditions and measurement techniques across laboratories can hinder the development of good-performing ML models. Additionally, human researchers can introduce bias that compromises the diversity and quality of data, for example, only successful experiment results are published and many details can be hidden [175, 176]. Another obstacle is that most of the databases available are commercial products that require a costly license, and programmatic Application Programming Interfaces (APIs) for large-scale data evaluation are rarely provided. Furthermore, the convenience of human readability present in a considerable portion of scientific publications poses a challenge to the accurate and convenient retrieval of information through automated means by computers. Despite this, innovative text mining techniques such as those presented in works [177, 178, 179, 180] show potential as a solution to this predicament. Finally, high-throughput and combinatorial experiments demonstrate a promising capacity to produce extensive experimental material databases with diverse properties in a short duration and are presently under active development [181, 182].

Given the limitations of experimental data, many researchers consider utilizing computational data sources to construct ML models. The idea of computational simulation has advanced across various scientific fields over the past century, encompassing a range of length scales from ab-initio techniques to finite element methods. Density Functional Theory (DFT) calculations, a widely-used Quantum Mechanical (QM) method in the field of physics and chemistry, have an advantage over experiments in terms of scalability as they can be more easily scaled across diverse chemical spaces. Especially due to the recent efforts of the Materials Genome Initiative [16], large public databases containing calculated material properties have spread worldwide. Some popular and well-known general-purpose databases with a high degree of property diversity are the Materials Project (MP) [15], AFLOWLIB [183], the computational materials repository [184], the Open Quantum Materials Database (OQMD) [185], the NOvel MAterials Discovery (NOMAD) repository [186], AiiDA [187]. These databases are usually based on extensive experimental records of inorganic materials like ICSD, complemented by a group of theoretically predicted stable structures, encompassing diverse properties such as relaxed structures, energetic profiles, electronic structure properties, and more. Some of these databases, including MP and AFLOWLIB, also incorporate well-defined APIs for rapid and comprehensive data retrieval, which is a fundamental need for efficient ML development [188, 189]. There are also many other specialized databases, such as the Harvard Clean Energy Project (CEP) [190] and the National Renewable Energy Laboratory (NREL) materials database [191], which have been used for several ML works which focus more on energy materials. In addition, many of these databases are equipped with open-source software platforms that boost data augmentation by conducting high-throughput computations based on the user's needs to better adapt ML models to diverse applications requiring more topic-specific data.

Beyond the quantity and quality of the data, the presentation format of data can also significantly affect the learning of an ML model. An ML algorithm typically learns more efficiently using a specific format, making the data representation of the material system of interest a critical consideration. The process of converting raw data into a more appropriate format (usually numerical values in the form of vectors or tensors, and they are commonly referred to as features, descriptors, or fingerprints in the literature) for an ML model to learn is known as feature engineering (featurization), which will be discussed in detail in a later section 3.1.2.

(iii) Model training

An appropriate model is selected based on the task's nature and trained on collected data. There are basically four types of ML tasks - supervised learning, unsupervised learning, semi-supervised learning (as well as active learning), and reinforcement learning. In materials science, the supervised learning task is the most common one. This type allows the model to learn a complicated function that connects input features, such as structure or composition of the crystal and molecular, to output properties (labels) such as energies and bandgaps [192, 193, 194]. For unsupervised learning, the goal of ML is to identify meaningful patterns from the data without relying on ground-truth labels. They are frequently employed as visualization techniques to aid in the analysis of high-dimensional data obtained from materials studies [195, 196, 197]. Semi-supervised learning and active learning methods have been utilized to predict the most promising candidates, with the goal of reducing the necessary number of experiments or simulations. For example, Bayesian optimizations-based active learning strategies have been employed to reduce the required training data [198] and improve performance by optimizing composition within a given chemical space [199]. In the case of reinforcement learning, models are often used to realize some kind of controlling agent (e.g., to control the operation of an instrument/device during experiments), however, they are still relatively less used in materials research because of the general lack of the comprehensive environment and equipment for fully automated experiments [200].

(iv) Evaluation of performance

According to the "no free lunch" theorem [201], no single ML model can consistently outperform others when averaged across all problems. The selection of ML models generally depends on domain expertise and begins with exploratory data analysis. For instance, utilizing unsupervised learning algorithms like PCA to conduct dimension reduction for enhancing visualization can aid in comprehending data distribution and inherent feature properties, leading to more effective model building. In practice, researchers usually need to try different ML models on the collected data and select the best-performing one based on evaluation metrics on the validation data set, such as least Root Mean Squared Error (RMSE), Mean Squared Error (MSE), and coefficient of determination (R^2) for regression task. Constructing an optimal model that accurately fits the available data, without overfitting or underfitting, is essential for researchers. Furthermore, it is crucial to carefully consider the hyperparameters that determine the model configuration and architecture.

On the other hand, although evaluating the performance of the model is critical, it is equally important to consider the interpretability of the model. In the scientific field, researchers typically prefer models that provide excellent intuitive interpretability and simplicity. These models can yield valuable insights and enhance the researcher's awareness of the matter at hand, consequently benefiting the development of innovative materials. The model's simplicity can reduce the overfitting problem and enhance computational efficiency, which is also beneficial for real-time applications. Researchers can also use feature importance analysis methods (such as tree-based models) to identify critical physical or chemical parameters that are related to predictive targets [202, 203, 204, 205], or to provide chemical intuition such as by visualizing hidden layer activations in deep learning models [206]. These findings can help researchers gain a better understanding of the behavior of the ML model in a more chemical/physical intuitive way.

(v) Application of the model

The well-developed ML model can then be used to help the development of materials. For example, desired candidate materials from a large pool can be selected before synthesis by predicting the properties of the unknown sample. Besides, ML models can be utilized to direct experimental/computational design, without the need for exhaustively performing expensive and time-consuming experiments or computations [48, 207, 208, 209, 210, 211, 212, 213, 214]. ML can also accelerate simulation studies, such as predicting energies or forces for a given atomic configuration using Machine Learning Inter-Atomic Potentials (ML-IAPs) in DFT simulations [194]. On the other hand, online platforms and services should be provided to users, which can facilitate easier accessibility to the developed ML models, and also allow for continuous updating of models when new data is added.

3.1.2 Featurization

The optimal representation of data depends on the objectives of the ML models and demands a thorough grasp of both the scientific problem in question and the workings of the learning algorithm. It is often unclear which representation can achieve the best performance, and this topic remains an ongoing pursuit [215]. Ideally, effective feature sets should offer sufficient information about the studied materials system for the target space to be predicted. In materials science, researchers usually need to perform featurization to identify robust features for representing materials. These features are often called fingerprints and they serve as the input for the ML model, whose task is to build a mapping function from these fingerprints to some properties of interest. The definition of fingerprints varies depending on the materials system, but a critical requirement is that the feature set should create a unique match with the crystal/molecular composition and/or structure of different materials at nano-, micro-, or macro-scales. Furthermore, it is also important to ensure that the feature set is not excessively elaborate or computationally demanding to facilitate efficient training as well as prediction and reduce the potential of overfitting. This is of particular significance in materials science, where data sets are often of limited size. Specifically, it is generally advisable to avoid incorporating redundant or highly correlated features. For example, when considering an element's properties, it may be most effective to select either its atomic number or its group and period number in the periodic table, as both contain comparable information. Additionally, many elemental properties, such as melting/boiling points and elastic constants, are highly correlated and are both influenced by cohesive energy [17].

There are many types of fingerprints available depending on the particular issue being examined and the accuracy requirements of the predictions. Fingerprints can be categorized based on different criteria, for example, they can be defined at different levels of granularity and can be roughly classified into three tiers when viewed on a scale from coarse to fine: gross level, molecular level, and atomic level [216]. Gross-level fingerprints are preferable for obtaining a high-level understanding of the factors behind complex phenomena, such as the mechanical strength of materials, especially when structure-property insights are of interest. Further, molecular-level fingerprints can provide insight into crucial atomic-level structural fragments and are vital for accurately predicting specific properties within a broad range of chemical materials, including but not limited to electrical properties and thermal conductivity. Taking another step forward, achieving even higher precision requires atomic-level fingerprints that contain structural information with sub-Angstrom resolution, allowing for extreme chemical accuracy, such as prediction of total energies and atomic forces in DFT simulations, and accurate identification of structural features, space groups, or phases.

From a different perspective, fingerprints can also be classified based on whether or not they include structural information of crystals or molecules, which refers to the positional information of the individual atoms and the bonds between them [17]. Composition-based features typically do not contain any structural information, and they solely comprise elemental information. Typically, the characteristics of materials are based on the properties of their constituent elements, including atomic number, atomic radius, density, electronic structure, electronegativity, etc. For example, Ward et al. [217] demonstrated that the use of these elemental physical properties as descriptors for materials can effectively predict various properties, including band gap energy and glass-forming ability. However, a significant limitation of composition-based features is their inability to distinguish between crystal polymorphs and molecular isomers/conformers, like diamond and graphite, which have vastly different physical and chemical characteristics. Composition-based fingerprints are typically limited to situations with constrained structural freedom, such as when focusing on specific prototypes like perovskite or garnets [218, 219], or when examining only the ground state polymorph [220, 221]. On the contrary, in many situations, spatial structure information proves to be vital. Therefore, it is preferred to possess a feature set that thoroughly describes the crystal or molecular structure. There are established guidelines that state the need for crystal and molecular structural features to have properties of translational or rotational invariance, and remain unaffected by the permutation of homonuclear atoms [222, 220]. The need for invariances can be tackled by using ML models such as convolutional neural networks. These techniques have exhibited the ability to address translation invariance to some degree. However, some cases like predicting potentials for molecular dynamics simulations could demand further restrictions to ensure differentiability for subsequent computations. Commonly used methods for featurizing molecular structures are: distance-based features like Coulomb matrix [223], London matrix [224] along with the Histograms of Distance, Angle, Dihedral (HDAD), and Molecular Atomic Radial Angular Distribution (MARAD) [215]; connectivity-based features like Simplified Molecular-Input Line-Entry System (SMILES) [225], Extended-Connectivity FingerPrint (ECFP) [226], and bag-of-bonds [227]; multiscale-combined features such as Atom-Centered Symmetry Functions (ACSF) [228], Smooth Overlap of Atomic Positions (SOAP) [222], bispectrum coefficients [229], and moment tensors [230].

In addition to all of the above, graph-based featurization has received considerable attention recently as a natural way to represent atoms (as nodes) and bonds between them (as edges) in molecules, resulting in state-of-the-art performance in predicting formation energies, bandgaps, and classifying metal/insulator [231, 232, 233, 234].

The community has developed various open-source software tools to support different representations of crystals/molecules, for example, RDKit [235], Dscribe [236], Matminer [237], Materials Agnostic Platform for Informatics and Exploration (MAGPIE) [217], MatErials Graph Network (MEGNet) [231].

It is often not enough to just build many candidate fingerprints/descriptors to represent materials. To construct a high-performing ML model, it is also crucial to select appropriate ones among candidate features. Feature selection is typically determined by domain-specific experts or through pure data-driven principles. Domain-knowledge-driven selection relies on researchers' physical and chemical intuition, which can introduce bias and lead to suboptimal performance or the omission of critical information. In contrast, data-driven methods start by generating a vast candidate set of features, which are subsequently refined by the model's performance. To identify promising features, researchers often begin with a limited number of primary descriptors and use them to construct numerous compound descriptors using algebraic combinations. Next, dozens of the most effective features will be selected from the large candidate pool. This down-selection is generally performed through automated means, for example, using L_1 regularization (i.e., Least Absolute Shrinkage and Selection Operator - LASSO [238, 239, 240, 241]) and its advanced variant Sure Independence Screening and Sparsifying Operator - SISSO [242], ordering feature importance [202, 243], genetic algorithms [244], compressed sensing [245, 246], and information science [247]. However, a significant disadvantage of employing data-driven feature selection is that the chosen features may not have a strong causal connection with the target variable and often depend heavily on the hyperparameters of the model [248].

Another method for feature selection is to utilize dimension reduction algorithms. These algorithms are used to synthesize new, low-dimensional features by combining the original high-dimensional ones, which can represent the most significant information in a more efficient manner, resulting in improved model performance. PCA [249] is one of the most frequently used approaches in this field [233]. In addition, manifold learning techniques [250] are also widely used as they can effectively capture intricate nonlinear relationships. For example, the autoencoder family [141], t-SNE [89], and UMAP [90], can learn extremely low-dimensional representations down to 2–3 dimensions. These algorithms are also widely applied in visualizing the elemental embedding vector trained from materials property [231], structural similarity [251], word embeddings in text mining of scientific literature [180], electronic fingerprints [252], and physical and genetic interactions visualization [253], etc.

3.2 Facilitation of materials development

In materials research, many ML studies aim to construct a model that efficiently maps the process-structure-property relationship of a material. This model enables rapid prediction of various quantities, such as compositions, structures, and properties of materials, which can later be applied to large data sets to quickly screen the most promising candidates for new material discovery, to optimize existing ones, or to guide the search process. The main idea of utilizing ML models in materials research is to predict the interested properties of the material system being studied in an efficient and accurate manner, ultimately replacing expensive experiments and simulations. Figure 3.3 illustrates this concept and the basic workflow.

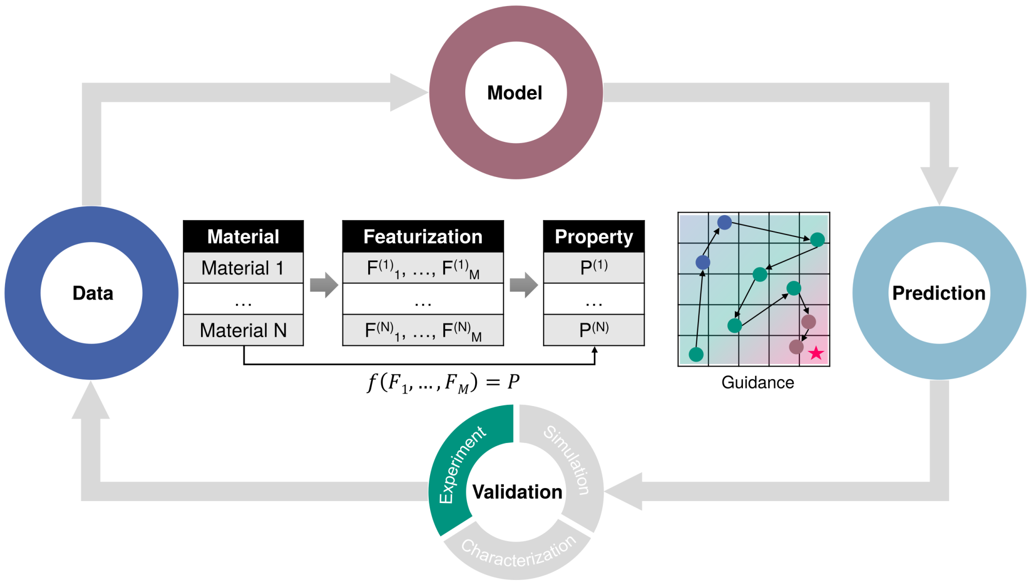


Figure 3.3: Concept of ML-based materials development, which works by building a fast predictive model between materials (represented in corresponding features) and their interested properties. The well-trained model can be used to explore and screen candidate space efficiently, leading to a reduced number of required experiments.

The following section presents a review of various application examples to illustrate the methodology of applying machine learning models in the development of different materials and their benefits. In section 4.1, this work shows a concrete example application to demonstrate the benefits of utilizing ML to accelerate the process of developing electrolyte materials in batteries. Specifically, the aim is to assist researchers in making efficient decisions during experiments, with the goal of reducing the number of attempts needed to find the optimal combination of parameters for synthesizing the best-performing material sample. This method concerns the optimization of the experimental parameters, which expands the range of application scenarios that mainly focus on searching for candidate materials in their compositional space.

3.2.1 Property prediction and discovery of materials

The discovery of new materials with high performance is a fundamental goal in the field of materials science. Currently, the process of identifying new materials involves a combination of experimental and computational evaluations, including the study of element substitution and structural modification. However, the current search spaces for composition, structure, or both are often severely limited, and standard screening methods will typically consume significant time and resources. Given the advantages of ML, it is being applied to the evaluation and screening of new materials "in silico" with the goal of providing suggestions for possible materials. There are two primary ML approaches to discovering new materials, which either focus on predicting composition or crystal structure [1].

For predicting composition, this problem can be reformulated as determining which chemical compositions are likely to yield compounds. Predicting stability is typically used to solve this problem and is often the key aspect in determining the stable existence of new materials. The formation energy, or energy above the hull of a material, is usually used as the metric for evaluating stability. Currently, DFT calculations have been demonstrated as an effective method for accurately estimating formation energies [254]. However, these calculations remain computationally expensive and are not suitable for traversing large combinatorial chemical spaces. The need for rapid prediction of formation energies (or closely related ones such as enthalpies of formation [255]) is in line with the application area of ML [218, 219, 256, 257]. Common applications include predicting perovskites and Heusler compounds, where the training data can be obtained from DFT simulations or experimental measurements [157, 158, 218, 219, 231, 232, 258, 259, 260, 261, 262, 263]. In addition to predicting the stability, there are also methods that use probabilistic modeling to directly predict the probability of a compound forming [220, 264, 265].

The prediction and characterization of crystal structures is another crucial topic for the efficient design of materials. If the crystal structure can be predicted in advance, it can help avoid unnecessary structure-related studies, such as significantly reducing the computational resource consumption during DFT calculation due to the enormous set of possible combinations. ML works mainly by learning empirical rules obtained from extensive data, which are highly effective in addressing this concern. Examples include predicting the probability of forming a particular binary crystal structure based on its elemental composition [266], predicting preferred crystal structure [241, 259, 267, 268], and classifying crystal structures [242, 269, 270].

Aside from discovering new materials, a common scenario arises when a material has already been discovered and is added to the database, but lacks measured or recorded properties. Prediction of the property of interest for this material is necessary in order to determine its applicability. To address this issue through ML, a common strategy is to construct a quick predictive model utilizing available data and map it to the database to screen for suitable materials [182]. This application has a wide scope, covering various multi-scale property predictions, such as vibrational free energies [271], defect energies [272], superconductivity [47, 48, 273, 274], melting temperatures [275], mechanical properties [257, 276, 277, 278, 279, 280, 281, 282], thermal conductivity [283], catalytic activity [284, 285], band gap [210, 243, 286, 287, 288, 289, 290, 291], different properties of polymer dielectrics [292, 293, 294, 295], to name a few.

Another area of interest is the design of new materials using deep generative models, such as Variational AutoEncoders (VAEs) and Generative Adversarial Networks (GANs) [296]. These models are particularly well suited for implementing inverse material design, wherein the desired material properties are initially listed, and then candidate materials are generated, validated, and evaluated on the fly [297]. Inverse design poses the challenge of exploring an extremely large chemical space and to solve this problem, generative models can be used to project the original chemical space into a low-dimensional space (acting as a dimensionality reduction), which can facilitate the optimization process. There, different materials can be represented as continuous low-dimensional variables, enabling simpler design and optimization of materials. Candidates found in the low-dimensional space can then be inversely mapped back to the original space and later verified for validity. Typical solutions include the use of (i) GANs to automatically suggest and evaluate novel molecules for a desired application [192] such as using CrystalGAN model [298] to generate, screen, and discover new stable hydride compounds for solid-state hydrogen storage applications, predicting new high-melting-point organic molecules based on Objective-Reinforced GAN for Inverse-design Chemistry (ORGANIC) model [299], discovering novel molecules for drug design and development [300]; (ii) VAEs to search for optimizing functional compounds such as drug-like molecules [301] in a targeted manner by performing gradient-based optimization directly in the latent space.

3.2.2 Adaptive design and active learning

The previous sections deal with the prediction of various properties. All of these methods inevitably have the ultimate goal of minimizing the time it takes to find a new optimal material with tailored properties. To achieve this goal, the number of experiments or computationally expensive calculations that need to be performed should be reduced, as these are the most time-consuming and expensive parts of the discovery process. ML is also beginning to gain popularity in solving this problem, which is usually referred to as adaptive design or surrogate-based optimization in this field [302, 303]. These approaches usually rely on uncertain information to direct the search process such as predicting the best experimental condition for synthesizing a material. Selecting the optimal candidate necessitates ML models that provide not only predictions but also associated uncertainty. In short, the region where the model produces the largest uncertainties usually indicates that samples from that region contain valuable information that can improve the model's performance. Therefore, it is worthwhile to explore.

Usually, this method involves an iterative updating process: the ML model generates the most promising sample, which is then validated by experiments or computations to evaluate its performance. The obtained results are then fed back into the model for re-fitting, leading to further improvement in performance. In the next iteration, the model will predict the next best candidate, and this process will be repeated until the optimal solution is found or the maximum number of attempts is reached. In order to decide which is the most worthwhile sample, an ML model usually needs to balance between exploration and exploitation [304, 305]. This involves exploiting the information in the existing data (i.e., samples with the best predictions) and exploring less sampled search space (i.e., data points that are likely to have large uncertainties). In the early stages of the iteration process, the goal of the model is typically to thoroughly explore and learn the search space. As the number of tested samples increases, the performance of the ML model over the whole search space is expected to improve gradually, and the associated uncertainties will decrease. An "adaptive design" scheme enables a gradual shift from exploration to exploitation, resulting in a step-by-step and progressive increase of training data that effectively searches for the target with minimal attempts.

In general, there are three primary methods for assessing uncertainty in the context of applying ML models to materials science. The first type of approach is the most popular one in recent years, which leverages the characteristics of probability distribution-based models to estimate the uncertainty of the prediction as well as the prediction itself. Bayesian methods are the most widely used models, among which the Gaussian process regression model receives the most attention [57]. This model assumes that the available data follow a Gaussian distribution. Consequently, the output of the model also follows a distribution instead of a scalar value, allowing its uncertainty to be measured in a straightforward way. This offers a significant advantage over other methods, which may require additional work at a significant cost to estimate uncertainties. The mean and variance of these estimates represent the most likely predicted value and the associated uncertainty of the prediction, respectively, which are inherent outcomes of Bayesian approaches [306]. The second type of approach uses a versatile scheme called bootstrapping method [307, 308], the idea of which is simple but practical: data fluctuations are artificially introduced (e.g., only a random portion of the data is used to train the model at a time) to train slightly different models, and the outputs of these models are collected to obtain uncertainty information. The mean and variance of the predictions generated by these models can serve as the prediction and provide information about its uncertainty. In essence, such an approach attempts to explore the sensitivity of the model to "perturbations" to the given data set. The third method quantifies uncertainty explicitly through the creation of heterogeneous models, which are models constructed with slightly varying architectures (or with different random initialization), such as using neural networks models with dropout layers [309] or decision tree-based models with different architectures, and the distribution of predicted values is used to estimate uncertainty [310].

The adaptive design has already been widely employed in materials science, for instance, some researchers use this method to discover, develop, and design targeted materials, such as NiTi-based shape memory alloys [199], polymers [311], solid-state batteries [312, 313, 314], solar cells [315], and many other material systems [316, 317, 318, 319, 320, 321]. It is worth noting that the process described above is sometimes referred to as "active learning" in the field of materials science. However, this differs somewhat from the original concept of active learning in ML, which is more focused on achieving a good-performing model by training with minimal data. In this case, the primary objective is to identify the best-performing samples with the fewest experimental trials.

Recently, the adaptive strategy has been also integrated with autonomous high-throughput experimentation performed by robots with on-the-fly decision-making [32, 322, 323, 324]. This integration holds promise for conducting ML-assisted exploration of vast material spaces with minimal human intervention. This could significantly accelerate the speed of materials discovery, while also assisting with research involving toxic materials or challenging environments, and mitigating human bias in materials studies [192, 325]. Some recent examples include the synthesis and crystallization of a new polyoxometalate compound [325], predicting the outcome of untested chemical reactions and identifying unusual reaction mixtures [326], automating experimentation for carbon nanotubes using a robot system [327].

3.3 Prediction of simulation

ML can accelerate computational simulations for complex material systems on larger length/time scales with high accuracy. This works by directly predicting simulation results or replacing parts of the expensive calculations, taking advantage of the speed of ML predictions while maintaining high accuracy. Predictions of simulations can be broadly categorized as predicting intermediate variables necessary for the simulation process, for instance, potentials in Molecular Dynamics (MD) simulations [30], Finite Element Method (FEM) simulations [328], or directly predicting simulation results, such as the structural evolution outcomes during Phase-Field Method (PFM) simulations [329]. Neural network models are commonly used in this field because simulated data is typically larger, richer, and more complicated than experimental data, often requiring a better-performing model with a higher degree of accuracy. Another special application is the use of neural network models to solve the numerical equations directly to obtain results instead of performing simulations. These ideas are presented in a unified manner and are illustrated in Figure 3.4.

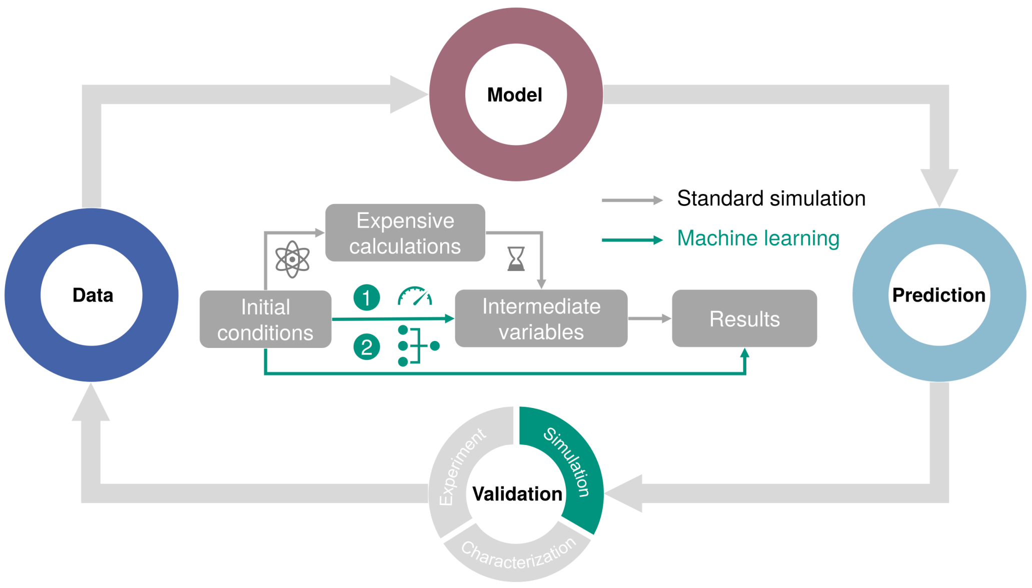


Figure 3.4: Schematic illustration of the principle of using ML to accelerate simulation: 1. replacement of some of the intermediate variables in the simulation process that are subject to expensive calculations; 2. direct prediction or solution of the final results of the simulation.

The following section presents a review of various application examples to illustrate the methodology and advantages of applying machine learning models in accelerating simulation studies. In section 4.2, this work illustrates an example to verify the effectiveness of the above concepts by building a fast prediction model for linking process-structure-property relationships in porous microstructures with a neural network model. This model can be used to predict the quantity that needs to be solved by computational simulation and can also be used to solve the inverse design problem efficiently in combination with other optimization strategies. In particular, this work leverages unsupervised learning to automatically learn basic structural features from large amounts of digitally generated microstructures, without the need for manual design of descriptive features. By combining the learned high-quality representation with transfer learning, it is possible to generalize to downstream tasks with only a small number of labeled samples, resulting in a fast predictive model for interested properties.

3.3.1 Acceleration of simulation

The fundamental challenge in computational materials science is the trade-off between scale (length and time), accuracy, and transferability. Applying ML to accelerate simulations generally involves using the rapid predictive capabilities of ML to replace the computationally expensive and time-consuming parts in the simulation, while maintaining acceptable accuracy.

One of the often-discussed topics is using ML to approximate potentials in MD simulations. Traditionally, semi-empirical potentials (classical force fields) based on physical rules are favored in MD due to their ability to access appropriate length and time scales. These potentials make use of prior knowledge about interatomic interactions under known conditions and use parameterized analytical equations to determine properties such as total energies and atomic forces. However, they are usually not transferable across chemistries and are of limited accuracy [30]. On the other hand, potentials based on quantum mechanical calculations can also be used in MD, e.g. those based on DFT calculations, are transferable and more accurate. However, they are usually computationally demanding for large systems and long-time scales. Hence, it is usually limited to runs of a few picoseconds and simulations with hardly more than thousands of atoms.

Recently, ML models have gained attention for bridging the gap between quantum mechanical and semi-empirical methods. These models offer surrogate models that combine the most favorable aspects of both methods, resulting in the development of ML potentials. Generating ML potentials is essentially a complex regression task that involves mapping from atomic configuration to properties like potential energy surface or the force field, or both, by fitting an ML model based on a large training data set of $\{$atomic configuration $\rightarrow$ property$\}$, typically containing thousands of high-fidelity DFT calculations. The prediction can be several orders of magnitude faster than normal calculation and it has the potential to maintain quantum mechanical and chemical accuracy, in particular potential energy and reaction enthalpy predictions with small errors < 1 kcal/mol, and atomic force predictions with errors < 0.05 eV/Å [216]. Other than predicting potentials for MD simulations, ML techniques can also be used to develop exchange and correlation potentials and energy functionals for DFT simulations [330, 331, 332] or bypass the problem of solving the Kohn-Sham equations (namely, directly learning the map between the potential and the density) [333], offering feasible, efficient, and fast ways to provide highly accurate simulation results.

The key challenge in achieving the aforementioned application is to develop appropriate fine-level fingerprinting schemes for the material system. These fingerprints need to meet specific requirements in simulations, such as maintaining strict invariance with respect to arbitrary translations, rotations, and exchanges of atoms, as well as being smooth and differentiable for small variations in atomic positions. Many candidate solutions have been proposed, including functions-based fingerprints [228, 334, 310], bispectra of neighborhood atomic densities [229], Coulomb matrices (and its variants) [223, 335], Smooth Overlap of Atomic Positions (SOAP) [222, 336, 337, 332], and others [338, 339]. In addition to this, highly accurate prediction is also very important, and numerous ML models have been utilized to accurately map the fingerprints to various material properties of interest, such as neural networks, GPR, and moment tensor [230]. One of the most successful and widely used schemes to date is Behler and co-workers' [228] approach, which employs symmetry function fingerprints that are mapped onto the total potential energy using a neural network. This has been used to study various systems including surface diffusion, liquids, phase equilibrium in bulk materials, etc. Furthermore, the combination of Bispectra-based fingerprints and GPR learning schemes has resulted in Gaussian approximation potentials [229], which have demonstrated good chemical accuracy, versatility, and efficiency. For a more comprehensive review of various fingerprinting schemes and ML potential/force field predictions, please refer to [340, 341].

Another type of application is more concerned with predicting the results of simulations directly, rather than the intermediate variables needed for subsequent simulations, such as predicting the structural evolution process in phase-field simulations [329, 342, 343, 344, 345, 346, 347, 348, 349] and building structure-property relationships [350, 351]. In some studies, a reduced-order representation of the original simulation domain is used for dimensionality reduction, resulting in higher accuracy. To learn the “evolution” process, ML usually needs to build a “dynamic” predictive function that can effectively model sequential (time-series) relationships, therefore, RNN-based models are the widespread choice. In other words, the researchers aim to be able to infer future outcomes based on the first few evolutionary steps of the simulation.

In addition, multi-fidelity learning [352] is also gaining popularity in accelerating simulations by bridging the gap between results obtained with less accurate but faster methods (low fidelity) and those obtained with slower but more accurate methods (high fidelity). The results derived from low-fidelity computations can be used as additional information to train machine learning models to predict their corresponding high-fidelity property values. This type of approach, known as multi-fidelity learning, is used in various domains to overcome computationally demanding engineering design problems [353], but has only recently emerged in the field of materials informatics [289].

3.3.2 Physics-informed neural networks

Purely data-driven ML models may fit observations (training data) accurately, but their predictions may be physically inconsistent or implausible due to extrapolation or observational bias. As a result, this can lead to inferior generalization performance. In contrast, another branch of ML applications integrates fundamental physical laws and domain knowledge into ML models. This provides useful prior information that includes robust theoretical constraints and inductive biases, in addition to observational evidence (training samples). A recent example of this learning philosophy is the family of Physics-Informed Neural Networks (PINNs) [354]. This approach is primarily geared toward conventional numerical domains, with a particular emphasis on addressing issues connected to Partial Differential Equations (PDEs), including equation solving, parameter inversion, model detection, control, and optimization. One of the main objectives of developing these algorithms is to enhance the generalization ability of ML methods that are more understandable and able to withstand imperfect data such as missing or noisy values and outliers. By utilizing prior knowledge and constraints, these methods can provide accurate and physically consistent predictions, even for extrapolation tasks [355, 356]. The primary use of PINNs is solving PDEs, specifically for Computational Fluid Dynamics (CFD) [357]. It is particularly interesting to note that for this particular application, PINNs are trained without any training data but only through initial and boundary conditions. The model then solves the equation by minimizing specifically designed loss functions for obtaining the results [354]. The principle of this process is to approximate PDE solutions by minimizing a specially designed loss function that incorporates physical constraints like residual terms of initial and boundary conditions and PDE residuals at specific points in the region (usually referred to as “collocation points”). After training, an inference can be performed to obtain spatio-temporal values at these points [358].

Examples of successful applications in materials development include identification and precise characterization of a surface crack in a metal plate [359], extraction of mechanical properties of 3D printed materials [360], analysis of internal structures and defects [361], design of composite materials [362], improvement of MD simulations [363], and many other systems [364, 365].

3.4 Enhanced analysis of characterization data

ML is a popular tool for analyzing and interpreting characterization data, which can provide valuable and intuitive insights. With the development of modern characterization techniques, the accuracy of measurement data increases rapidly and the amount of data explodes, which renders traditional analysis methods obsolete and time-consuming. For instance, interpreting and labeling experimental images (such as Scanning Transmission Electron Microscopy - STEM) [366] and spectra (such as X-Ray Diffraction - XRD) [367] typically demand significant expertise and extensive time investments from researchers to accomplish. ML models adapted from other application areas, including factor analysis, computer vision, and speech recognition, can be advantageous in simplifying analysis and expediting the process. Based on the data dimensions, characterization data can be categorized into spectral data (usually 1D) or image data (usually 2D or 3D). The concept of applying ML in analyzing and interpreting characterization data is shown in Figure 3.5.

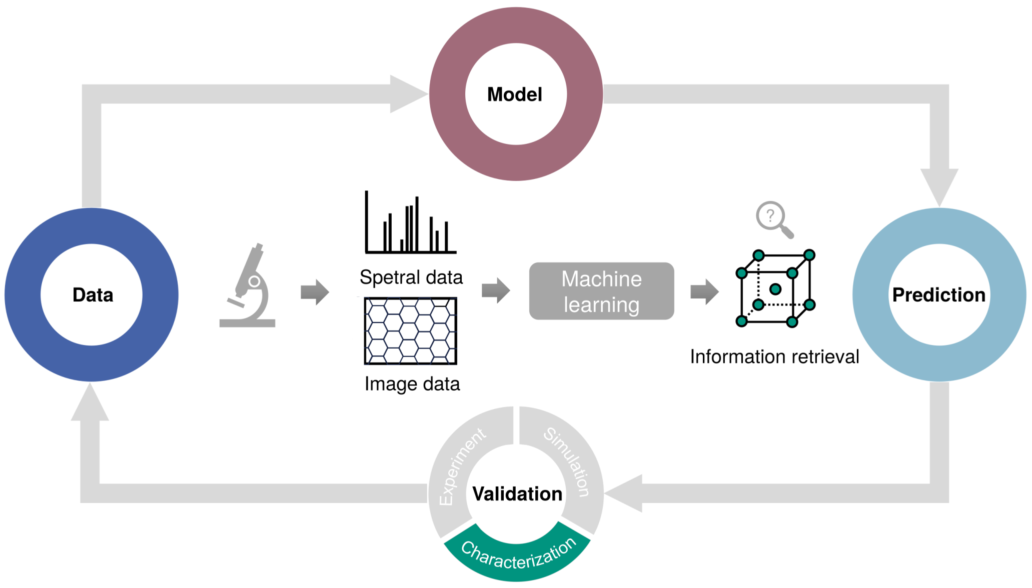


Figure 3.5: Schematic illustration of applying ML in analyzing and interpreting characterization data for better information retrieval from spectral data or image data.

The following section presents a review of various application examples to illustrate the methodology of applying machine learning models in analyzing various kinds of characterization data. In section 4.3, this work demonstrates a robust analytical method that enables researchers to analyze information from measured spectral data of lithium-compound materials in an intuitive way. A robust workflow is proposed and the study comprehensively compares the results of different machine learning methods based on a limited amount of data. In particular, the original problem is analyzed from multiple points of view and modeled from both classification and regression perspectives, breaking the traditional single modeling approach. The developed models can assist researchers in recognizing different lithium compounds and identifying compositions from mixture samples. In addition, chemical information that corresponds to the properties of the materials can be obtained through further analysis of the model, enhancing their interpretability and validity.

3.4.1 Spectral data

Spectroscopy is a field of study that examines the spectra of electromagnetic radiation as a function of wavelength or frequency using spectrographic instruments and other methods to learn about structural, dynamic, and compositional properties of materials. Spectroscopy techniques, therefore, serve as a critical tool for characterizing materials. Spectroscopic analysis usually involves fitting a quantitative physical model, for example, Rietveld refinement for XRD, or more empirical approaches, such as fitting linear combinations of reference spectra, as is the case with X-ray Absorption Near-Edge Structure (XANES) [368]. Typically, these approaches require considerable expertise and proficiency of the researcher. In recent years, the increasing use of advanced instrumentation and high-throughput experiments, along with the surge in data amount, has presented several challenges for dealing with the "big data" scenario. However, these advances also provide opportunities for researchers to use ML techniques to automatically extract knowledge from the data, potentially reducing the repetitive and time-consuming work involved in the analysis process.

In the analysis of spectral data, ML is commonly employed to aid in the extraction of information, specifically crystal structure and mixture components. The most prevalent usage is in XRD, a well-established technique for characterizing the crystal structure of materials. Because ML models typically require large amounts of data for training, the majority of research in this field has centered around the use of extensive simulated XRD data. Usually, researchers are interested in extracting or predicting structural information from XRD spectra, such as space group prediction [367], crystal classification [369, 370], phase identification from samples consisting of a mixture of several phases [371], and others [372]. ML usually requires data cleaning and processing, including duplicate checking. Therefore, it is crucial to give extra attention to the XRD data, considering that the database (such as ICSD [373]) may contain many duplicates that have to be filtered out to avoid information leakage for a fair evaluation. Furthermore, the problem of imbalance in the types of structures in each space group can also lead to difficulties. Additionally, in order to improve the model's performance on XRD data, researchers have also designed several special data augmentation methods to take into account the nature of XRD, such as adding extracted experimental spectral noise to the calculated XRD [374], using an ensemble of models to reproduce experimental variations (missing peaks, broadening, peak shifting, noise) [375, 376], or proposing a modular neural network architecture to reduce overfitting that allows the combination of diffraction patterns and chemical data to provide a ranked list of predictions [377]. Some successful examples include the determination of structural parameters [378], classification of materials based on phases or structural properties [379, 380], automation of phase diagram generation using high-throughput XRD measurements for various compositional ranges [20, 381, 380, 382], and interpretation of multiphase [383].

In addition to XRD, ML methods have also been applied to other spectral data, for example, to analyze the structural information on different scale levels. This includes predicting coordination numbers from X-ray Absorption Spectroscopy (XAS) [384, 385, 386, 387, 388], discriminating and estimating the concentration of components in a mixture from vibrational spectroscopies such as Raman and InfraRed (IR) waves [389, 390, 391, 392, 393, 394, 395], classifying space groups from Pair Distribution Function (PDF) data [396].

On the other hand, the inverse problem of the above, i.e., predicting the spectroscopy itself, where the data can be viewed as a high-dimensional material property of the structure, is also interesting. This is common in molecular science, where the predictions of infrared spectra [397] and molecular excitation spectra [398] have attracted wide interest. Examples are prediction of infrared spectra directly from molecular structure descriptors using neural networks [399, 400], prediction of phonon Density Of State (DOS) spectra [401] from atomic positions and element types [402, 403], and prediction of XAS spectra [404, 405].

In addition to learning structure-spectra or spectra-structure relationships, some studies have also explored the possibility of relating spectra to some other material properties, such as predicting catalytic properties from electronic DOS spectra calculated from DFT [406], mapping between the image and the UltraViolet-visible (UV-vis) spectrum of the material [407], and predicting gas adsorption isotherms [408, 409]. In addition to prediction problems, ML methods can also be used to assist in the analysis of complex characterization data. Many MultiVariate Analysis (MVA) techniques [410], such as PCA, and Non-negative Matrix Factorization (NMF), are showing great promise for analyzing complicated high-dimensional spectral data. Among them, NMF is a method closely related to PCA in that it takes a set of patterns as a matrix and then compresses the data by reducing the dimensionality by finding the most important components. In NMF, a constraint is applied that all the components and their weights must be strictly positive. This constraint often corresponds to many real physical situations, for instance, spectra typically possess positive values, as do the weights (amounts) of chemical elements. As a result, NMF can often result in decomposed components and weights that are more interpretable and physically meaningful. Examples include decomposing mixture spectra to recover individual components [411], and analyzing multi-component systems such as Time-of-Flight Secondary Ion Mass Spectrometry (ToF-SIMS) [412].

3.4.2 Image data

With numerous image-based techniques available for material characterization, researchers have used a variety of Computer Vision (CV) techniques, resulting in a wide range of applications. Image data from characterization techniques often span multiple length scales and contain information about local atoms, defect distribution as well as type, material composition, microstructure, etc. Technological advances in characterization facilities have led to an exponential increase in the number of image data sets, requiring an automated high-throughput method to analyze these large and complex data, ideally in real time. Good overviews of ML applications in different characterization techniques can be found in papers [413, 174].

According to the needs in image analysis, the applications can be roughly divided into image classification, prediction of properties of interest, object detection, instance or semantic segmentation, image enhancement to facilitate further analysis, auto-tuning of experimental imaging parameters, microstructure representation learning, and so on. The following are some examples of typical applications.

Classification and regression are the processes of predicting discrete or continuous variables of the subject being photographed. For classification applications in image data, researchers can classify different material systems such as particles and fibers in Scanning Electron Microscope (SEM) images [414], detect cracks in macro-scale materials images [415, 416], identify the space group of simulated diffraction patterns for crystal structures [270], classify symmetries in simulated scanning tunneling microscope measurements of 2D material systems [417], predict the corresponding space group for a given Electron BackScatter Diffraction (EBSD) pattern [418], etc. For regression applications, ML models can be used to predict interested property on noisy and heterogeneous industrial data sets of limited size [419, 420], predict the Euler angles representing the orientation in EBSD [421, 422], learn crystal plasticity from images of strain profiles [423], predict the strain distribution across a given 3D microstructure [424], and many other systems [425, 426].

In addition to classifying or predicting certain quantities from an image, object detection techniques can be used to detect/localize individual instances of objects in the image. The traditional approach is to manually design features to anchor the target object that relies heavily on expert knowledge such as setting thresholds, edge detectors, or other hand-crafted filters, which cannot effectively deal with noise and other artifacts in the data, making manual analysis time-consuming or even impractical. The ML approach, on the other hand, works by providing the model with a large number of example detections and making the model learn its own criteria to perform the task. This technique has been used extensively to detect individual atoms and defects in microstructural images of various material systems, including detecting atoms in simulated atomic resolution TEM images of graphene for autonomous defect characterization [427], detecting and tracking atomistic defects in sequences of atomic resolution STEM images of WS_2 [428], detecting vacancies and dopants from STEM images [429], identifying defects in transmission and STEM images of steel including dislocations, precipitates and voids [430], and so on [431, 432, 433, 434].

Another closely related field to target detection is image segmentation. This process involves recognizing objects at the pixel level (in other words, classifying each pixel). Pixel-wise image segmentation is believed to be able to provide more detailed information about the size, position, orientation, and morphological features present in images. It has received much attention in materials science research, such as instance-level segmentation of individual particles and satellites in dense powder images and further acquirement of morphological statistics [435], segmentation of martensite, bainite, and pearlite in SEM images of carbon steels [436], segmentation of microstructural constituents in the ultra-high carbon steel database [437], characterization of dendrite growth from computed tomography data [438], and segmentation of components in lithium-ion batteries [439].

ML can also be used to improve the quality of image data, facilitating further analysis and reducing the time required to acquire information from images. For example, super-resolution techniques can be used to artificially increase the resolution of TEM/SEM images, which require long dwell times and can introduce undesired artifacts during high-resolution image acquisition. Successful examples include mapping low-resolution SEM images to higher resolutions for carbon samples containing gold nano-particles [440], reconstructing full STEM images from corresponding partial images [441], and enhancing SEM imaging of lithium Nickel Manganese Cobalt (NMC) oxide particles [442]. Besides, ML can also be used to reduce noise such as for tomography images [443], benefit further analysis by data augmentation through style transfer [444], generate synthetic image using GANs [445], colorize images [446], refocus SEM images [447], and so on.

ML is also employed for automating or improving the process of capturing images, such as automating the tip conditioning for Scanning Probe Microscopy (SPM) experiments, i.e., monitoring artifacts resulting from degradation in tip quality and automatically re-conditioning the tip [448], autonomously correcting the defects and adjusting parameters during manufacturing for 3D printing [449], detecting chemicals, materials, and transparent vessels in a chemistry lab setting for autonomous experimentation [450].

3.5 Information extraction from literature

In addition to accelerating the process of developing materials, ML can aid researchers in rapidly extracting valuable information from extensive documentation, which can considerably diminish manual labor. Although numerous scientific literature can offer researchers a wealth of information, navigating through it is becoming increasingly difficult due to the abundance of journals, publications, and databases. Text mining, a branch of ML techniques, has emerged as a favored approach for identifying and extracting data from unstructured text sources such as papers. This concept is illustrated in Figure 3.6.

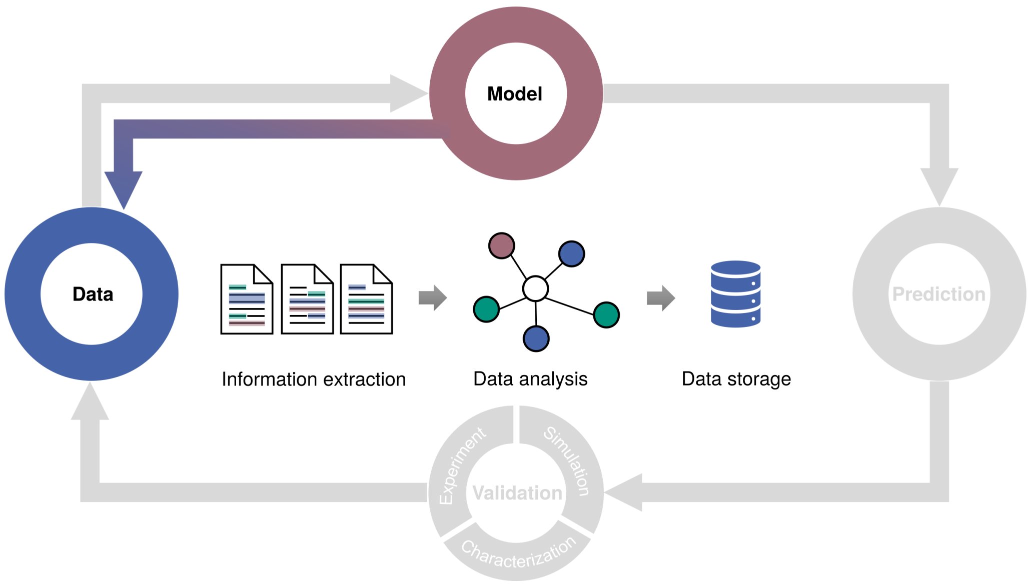


Figure 3.6: Schematic illustration of using ML to perform text mining from scientific publications for extracting structured, useful information.

Text mining makes use of a variety of sophisticated and specialized techniques, such as a combination of traditional text processing and Natural Language Processing (NLP) techniques. This method can be used to transfer knowledge between different scientific disciplines/domains, analyze high-throughput experiments, develop specialized databases, and extract logical facts and relationships in a structured form [179]. The extracted information can be utilized to aid researchers in making objective inferences within general decision-making contexts, including the design of experiment [451].

A large portion of the current knowledge in the field of materials lacks structure and is only presented in the form of unstructured text, tables, or images in various publications. Ideally, it is recommended to convert publications from a document format, often PDF, into a machine-readable format, such as HyperText Markup Language (HTML) or JavaScript Object Notation (JSON), for easier parsing. In general, using text mining techniques to analyze text data involves several steps, including retrieving the raw data, performing appropriate forms of "preprocessing" such as tokenization, word stemming, and lemmatization, various forms of feature extraction such as word embedding as vectors, and finally train ML models for information extraction. There are many challenges in this area, including resolving long-term dependencies between words and phrases across multiple sentences and paragraphs during the analysis of very long text content. Besides, copyright restrictions can often present challenges as access to or extraction of various scientific data sets, such as peer-reviewed scholarly articles and patents, are often limited. Another challenge in extracting text information stems from the specialized nature of materials science, which involves complex chemical formulas, scientific terminology, and notations like oxidation states or symbols for physical units, as well as mathematical equations. These factors may contribute to erroneous parsing of data. To solve this issue, specially designed software is needed to perform good text preprocessing to enhance the identification of chemical formulas and elements. Several software libraries are under active development in this community, for example, ChemDataExtractor [178], ChemicalTagger [452], MaterialsParser and Borges [453, 454], and mat2vec [180].

NLP methods have also been applied to extract information, search, and discover materials. One of the primary applications of NLP techniques involves retrieving data sets from published research texts. Collecting and retrieving data from literature has typically required researchers to possess extensive knowledge and experience, as the process involves manually entering information to create a complete data set, which can be both laborious and time-consuming. Successful examples of applying NLP to extract data sets include Curie magnetic phase transition temperatures [455], battery properties [456], UV-vis spectra [457], and surface and pore characteristics of metal-organic frameworks [458]. In recent years, DL-based NLP approaches such as the use of RNN-based models and transformer-based models, are becoming popular and they have been used to extract various categories of information [459], in particular, materials synthesis information [453, 454, 460]. The mined data can be further used to predict synthesis routines for titania nanotubes [179], different binary and ternary oxides [461], and perovskites [462]. Additionally, the codified databases can be further used to train other ML models to screen materials with target functional properties, such as the discovery of $HoBe_2$ with large magnetocaloric properties [463], and the design of dye-sensitized solar cells [464]. Another interesting use case involves predicting the suitability of untested materials for specific applications, such as identifying potential thermoelectric materials by calculating similarities between their compositions and application keywords [180]. Additionally, there is potential to explore under-researched areas of metallocene catalysis through similar predictive methods [465].

More recently, Large Language Models (LLMs), which can systematically analyze natural language and possess a good understanding of human language text, have been utilized for analyzing publications [466]. Their typical applications include generating formatted information such as tables or structured templates. For instance, the fine-tuning of a GPT-3-based LLM enables the direct conversion of scientific text to structured tabular data stored in JSON format [467]. To facilitate data extraction from literature by LLMs, there are two primary strategies: utilizing documents as external databases for LLMs to query (more general-purpose information retrieval), or retraining LLMs on collected data sets to enhance their accuracy (better performance for specific domains or tasks). BERT [468] and GPT [143] are two famous branches of LLMs, based on which new possibilities for extracting information from literature are emerging. Examples include MatSciBERT [469], Battery-Bert [470] and large polymer corpora [471]. Among LLMs, ChatGPT, an AI chatbot developed by OpenAI and released in late 2022, has perhaps received the most attention for its crushingly competitive performance compared to other models (although it is a commercial, non-open source one), and it has been used to extract scientific information from literature [472, 473] or to act as a research assistant [474]. Such models have the potential to greatly accelerate the research process by helping researchers extract the data they need from the vast amount of literature and breakthroughs are expected in the coming years to disclose data and knowledge currently hidden in scientific publications through the utilization of these techniques.

Chapter 4

Results of machine learning applications in materials science

This chapter presents three example studies that demonstrate the advantages of using ML to facilitate materials research. These studies cover typical application scenarios in materials research, namely, experiment, simulation, and characterization, and illustrate how ML methodologies can be applied in solving different problems from predicting interested properties to analyzing complicated data in the materials studies. In the first study, an optimization algorithm is used to design the experimental parameters for the sintering synthesis of electrolyte material in a solid-state battery. The goal is to minimize the time and trials required to find the optimal experimental parameters. Similarly, this approach is extended to determine the optimal parameters for generating digital porous membrane microstructures. In the second study, a neural network-based model is used to develop a rapid prediction model to establish the process-structure-property relationships for digitally generated porous membrane microstructures. The third work focuses on the analysis of characterization data measured from the time-of-flight secondary ion mass spectrometry of lithium-compound materials, where different machine learning techniques are investigated and compared.

4.1 Design of experiment for electrolyte in battery

Lithium-ion batteries with solid electrolytes provide greater safety, energy density, and long-term performance, representing a promising alternative to traditional liquid electrolyte batteries. One potential solid electrolyte candidate is Lithium Aluminum Titanium Phosphate (LATP), given its high Li-ion conductivity. To systematically evaluate LATP's performance, it is necessary to investigate the influences of experimental parameters on materials design. In this section, a materials design approach utilizing Machine Learning (ML) is used to design experimental conditions for synthesizing LATP. Experimental parameters are varied with a focus on studying the tolerance toward possible deviations in precursor concentration, sintering temperature influence, and holding time. Specifically, the study compares models created using diverse design selection strategies, based on training data collected from previous laboratory experiments. The optimal model is then selected to design new experiment parameters, followed by measuring the performance of newly synthesized samples. To gain a deeper understanding of the mechanisms behind the high ionic conductivity displayed by these samples, the resulting phase compositions and crystal structures are analyzed using X-ray diffraction. In addition to this, the microstructures of sintered samples are also investigated using scanning electron microscopy. This research illustrates the benefits of utilizing ML to design experimental parameters for synthesizing desired materials, which can effectively reduce the number of required experiments. These results have been been published in [475] and the following sections are based on this publication. The overview of this study is shown in Figure 4.1.

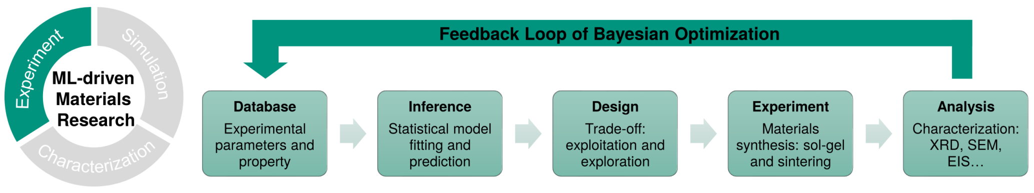

Experiment
Simulation
Characterization
ML-driven Materials Research
Feedback Loop of Bayesian Optimization
Database
Experimental parameters and property
Inference
Statistical model fitting and prediction
Design
Trade-off: exploitation and exploration
Experiment
Materials synthesis: sol-gel and sintering
Analysis
Characterization: XRD, SEM, EIS…

Figure 4.1: An overview of ML-assisted design of experiments by applying the Bayesian optimization method. It consists of five main components and forms an iterative loop: (i) collecting initial data points from experiments; (ii) fitting an inference model (e.g., a Gaussian process regression model) based on the existing data set; (iii) predicting the property in the search space along with uncertainty, taking both prediction (of the best known so far) and uncertainty (i.e., exploitation vs. exploration) into consideration for selecting the next optimal experimental configuration which has the potential to yield a better-performing material sample; (iv) performing the experiment to synthesize new sample and (v) validating the new sample's performance using different characterization techniques. The resulting sample is fed back into the initial training data set for the next iteration. Reproduced with permission [475].

4.1.1 Materials and data

Currently, Lithium-Ion Batteries (LIBs) are widely used, as they show great promise as an effective energy storage technology for a wide range of applications from mobile devices to electric vehicles. However, commercial LIBs confront hidden risks which are due to the utilization of fluid electrolytes, which may cause a variety of safety and performance problems, such as the potential ignition of the flammable solvent. To address these problems, lithium-ion batteries with solid electrolytes have the potential to be safer and longer-lasting alternatives with higher energy density compared to conventional liquid electrolyte batteries by allowing the use of high-voltage cathodes, which can decrease flammability, and suppress dendrite formation [476]. However, the principal challenge of solid electrolytes is their restricted ionic conductivity, which is typically many orders of magnitude lower than that of liquids ($1 \times 10^{-2}\,\mathrm{S\,cm^{-1}}$) [477]. The feasibility of these concepts depends on the applied solid-state electrolyte, for which a wide range of materials is being considered [478]. One of the promising materials is the family of lithium-containing NASICON (sodium superionic conductor) materials, such as LATP, $Li_{1+x}Al_xTi_{2-x}(PO_4)_3$ [479]. They have received wide attention due to their high ionic conductivity, low cost, and stability [480].

The ionic conductivity of LATP is particularly high for the composition $Li_{1.3}Al_{0.3}Ti_{1.7}(PO_4)_3$, and several studies have reported values up to $1 \times 10^{-3}\,\mathrm{S\,cm^{-1}}$ [481, 482, 483, 484]. $Li_{1.3}Al_{0.3}Ti_{1.7}(PO_4)_3$ ceramics have been successfully synthesized by different routes, such as calcination of stoichiometric mixtures of oxide precursors [485], glass crystallization [483], or sol-gel [484]. However, these studies are usually limited to laboratory scale, i.e., in small quantities and under experimental environments. In order to make ceramic electrolytes usable and competitive in the next generation of batteries, it is necessary to identify processing routes for upscale production. The sol-gel route has already been adopted successfully for the mass production of many materials for industry and therefore provides a good basis for LATP synthesis. For the sake of maintaining quality and reproducibility, it is necessary to identify the critical processing parameters. As the first step towards the upscaling of LATP production via the sol-gel route, the influence of possible deviations in the concentration of the precursors was investigated in a previous study [486]. This applies especially to phosphoric acid which is difficult to specify due to its hygroscopicity. In the case when precursors are not exactly stoichiometric, this can easily lead to the second-phase formation. In particular, for LATP, such second phases have a great influence on the densification and ionic conductivity. Hupfer et al. [487] report how the second phases $AlPO_4$ and $LiTiOPO_4$ can have impacts on the properties of LATP. In this work, the synthesis of LATP is studied by varying concentrations of the reactants, dwell time, and sintering temperature while microstructures, phase compositions, and ionic conductivities of the samples are further analyzed.

A key challenge in developing better materials is the large potential search space for optimal chemistries and processing conditions. Traditionally, the development of new materials requires a vast number of experiments guided by intuition, trial, and error, and is complemented by simulations and other tools to analyze the mechanism or optimize the design [488]. As a result, this process is time-consuming and challenging, which is often accompanied by detours or serendipity. Recently, the use of machine learning methods to accelerate materials development has received a lot of attention and many advances using this kind of technique have been made in the study of solid-state electrolytes, such as screening fast ion conductor candidates in supervised [214] or unsupervised [489] manner, filtering electrolytes in consideration of suppression of dendrite formation in lithium metal anodes [490], and developing good candidates combining theoretical calculations as well as experimental data sets [491].

In this study, LATP powders are prepared following a modified sol-gel synthesis route described by [484]. Appropriate amounts of lithium acetate $Li(C_2H_3O_2) \cdot 2\,H_2O$ (purity $\geq$ 99%, Alfa Aesar GmbH & Co KG, Germany), aluminum nitrate $Al(NO_3)_3 \cdot 9\,H_2O$ (purity $\geq$ 98.5%, Merck KGaA, Germany), titanium-isopropoxide $Ti[OCH(CH_3)_2]_4$ (purity $\geq$ 98%, Merck KGaA, Germany) are used as precursors. Lithium acetate and aluminum nitrate are dissolved in distilled water under constant stirring. Titanium-isopropoxide is then added dropwise to the solution. By adding the phosphoric acid slowly through a drip funnel, a white gel forms, which is then dried at room temperature for 24 hours. The subsequent heat treatment is performed in two steps: first, samples are heat treated at 400 °C for 6 hours to achieve the precursor formation and to eliminate reaction gases; second, samples are then processed at 900 °C for 8 hours to complete the reaction to crystalline LATP. One sol-gel batch is prepared with all precursors in stoichiometric quantities (marked as 0.0 wt%). To test whether the sol-gel route is tolerant against possible deviations in the concentration of the precursors, different sol-gel batches are also explored with either an excess up to +7.5 wt%, or a deficiency up to −15.0 wt% of phosphoric acid compared to the stoichiometric composition.

To ensure a high sinterability, the obtained powders are further processed in a planetary ball mill. The pellets are formed by uniaxial pressing and then further densified by cold isostatic pressing at 400 MPa. All pressed samples have a green density of approximately 62 % relative density. Samples are sintered at temperatures ranging from 850 to 1050 °C and with isothermal sintering time between 30 and 540 min. After sintering, samples are cooled down to room temperature in a furnace, and their corresponding densities are determined by Archimedes' method. For the ionic conductivity measurements, impedance analysis is performed at room temperature over the frequency range from 0.1 Hz to 1 MHz with an AC amplitude of 50 mV in the frequency response analyzer (AMETEK GmbH, VersaSTAT 4, USA). For further details of the experimental part please refer to the previous work [486].

To apply machine learning methods in guiding the experimental study, 80 initial data points from the aforementioned phosphoric acid deviation study are used to train the Gaussian Process Regression-based Bayesian Optimization (GPR-BO) model. These data points are sampled from an equidistant grid from −22.5 wt% to +7.5 wt% deviation of phosphoric acid compared to the stoichiometric composition. Sintering parameters

Table 4.1: Search range of experimental parameters.

	Iteration 1–2	Iteration 3–4
Experimental parameters	Value range	
Rel. amount of acid wt%	-22.5, -15.0, -7.5, 0.0, 7.5	-22.5, -18.75, -15.0, -11.25, -7.5, -3.75, 0.0, 3.75, 7.5
Temperature °C	800–1100 (step size 50)	800–1100 (step size 25)
Time min	10, 20, 30, 40, 50, 60, 90, 120, 240, 360, 480, 540	10, 20, 30, 40, 50, 60, 90, 120, 150, 180, 210, 240, 270, 300, 330, 360, 390, 420, 450, 480, 510, 540

with temperatures ranging from 800 °C up to 1100 °C in steps of 100 °C and isothermal durations of 10, 30, 60 and 480 min are applied. To further investigate the effect of these synthesis and sintering conditions on the properties of LATP, the machine learning model is used to predict promising candidates to investigate. Considering the long time needed to synthesize samples with different acid concentrations, the experimental parameters available to the model are expanded in two steps: for the first two iterations (1–2), the model is only allowed to make a choice among the available experimental settings; for the last two iterations (3–4), the range of allowed acid concentrations is expanded. Such kind of condition setting is derived from the results of previous grid search study and offers an efficient compromise that would address the otherwise excessively large search space. In total, 22 new samples have been synthesized in 4 iterations following the model's predictions. Detailed settings for the experiments are listed in Table 4.1.

4.1.2 Choice of Bayesian optimization strategy

Bayesian Optimization (BO) based on the Gaussian Process Regression (GPR) model is used to design experimental parameters in this work. It is a class of machine-learning-based optimization methods that focuses on solving the problem $\arg\max_{\boldsymbol{x}\in\mathcal{X}} f(\boldsymbol{x})$ within a domain $\mathcal{X} \subseteq \mathbb{R}^D$ as the bounding box. Its ability to optimize expensive black-box derivative-free functions makes BO extremely versatile [93]. Recently, it has become extremely popular for tuning hyperparameters in machine learning algorithms [94] and for the design of experiments in materials science.

In order to make full use of our experiment facilities, it is better that the model can suggest several samples simultaneously in the pre-defined search space in each iteration of BO. However, one of the limitations of BO is that the acquisition is myopic and permits only a single sample per iteration [492]. To alleviate this problem, the so-called Kriging believer approach [493] is used to suggest 5–6 samples at the same time during each iteration of BO in this study: to suggest more than 1 sample in each iteration, this approach (temporarily) adds each predicted sample to the training data set for updating the model, and then predicts another sample subsequently.

As there is no clear indication of which optimization strategy to use (according to the "no-free-lunch theorem" [494], there is no universal best-performing model for all problems), optimization efficiencies of BOs are compared with random search, Expected Improvement (EI), Upper Confidence Boundary (UCB) and Probability of Improvement (PI) strategies with the 80 initial points from the grid search study. The strategy of achieving the maximum ionic conductivity ($1.09 \times 10^{-3}\,\mathrm{S\,cm^{-1}}$) with the least number of average iterations is considered to be optimal. During the experiment of comparing these strategies, random noise is added to the observations, and the sample is allowed to be picked more than once (namely, with replacement). In detail, a given number of samples is randomly selected from the training data with replacement as initial data points, then the model is trained using a given acquisition function and the total number of extra tries (after initial random picks) needed to find the best sample (that is, the one with the largest ionic conductivity) in the grid search study is counted. The model is only allowed to make up to m attempts ($m = 80 -$ number of initial data points) to find the maximum value. This process is called a "virtual experiment" and is repeated 200 times with different sets of randomly selected samples. In the overall count, the initial random picks are excluded. Detailed results of comparing different strategies of acquisition functions are shown in Figure 4.2.

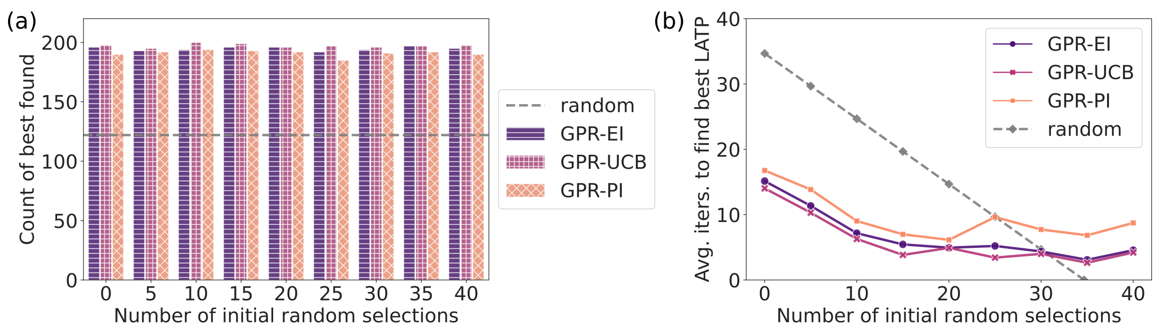


Figure 4.2: Comparison of GPR-based Bayesian optimization with different acquisition functions, EI (dark purple), UCB (light purple), PI (orange), and random selection (grey dashed): (a) number of counts to find the global maximum in the repeated 200 *virtual experiments, where random selection (grey dashed line) can find* 126 *times* (63%)*; (b) number of additional tries of different strategies needed after n initial random selections (x-axis) to find the global maximum of ionic conductivity* ($1.09 \times 10^{-3}\,\mathrm{S\,cm^{-1}}$) *in the training data set (grid search). On average random selection takes about* 34.7 *tries to arrive at the maximum. It is noticeable that only those cases where the global minimum is found are counted to calculate the average. Reproduced with permission [475].*

Figure 4.2a illustrates how many times different strategies can find the global maximum

($1.09 \times 10^{-3}\,\mathrm{S\,cm^{-1}}$) within the 80 tries in 200 virtual experiments. The grey dashed line represents how many counts the random selection can find the global maximum. Among the 200 virtual experiments, the random selection can find 126 times (63%) and it acts as the baseline for evaluating the performance of other models. From the figure, it can be seen that the GPR model with an EI (dark purple) or UCB (light purple) acquisition function can find the maximum in most virtual experiments ($\geq$ 95%), where UCB performs slightly better than EI in some cases. In contrast, PI (orange) performs slightly worse than the other two, but still much better than the random selection. We speculate that PI sometimes gets stuck in the local optimum, making it difficult for the model to reach the global maximum. As a result, it fails to find the global maximum in some cases.

Figure 4.2b illustrates the average number of extra tries (after a given number of initial data points) required for the models with different acquisition functions to find the global maximum. The random selection takes an average of 34.7 trials to find the global maximum, which is marked as a grey dashed line in the figure. It can be seen that all three models perform much better than the random selection. The performance of EI and UCB is similar, with UCB being slightly better and it takes the fewest extra tries to achieve the best result. More specifically, the number of extra tries needed to find the global maximum for EI and UCB decreases quickly with more initial data points, which is reasonable as the models' ability to fit and predict is gradually enhanced. After more than 15 initial data points, the gain of introducing more initial data points gradually decreases and the required extra tries finally stabilizes at about 5. This phenomenon is very beneficial for experiments because the model can achieve good performance even with only a small number of initial data points, significantly reducing the number of attempts required. In contrast, the performance of PI is worse than the other two strategies as it requires more steps to obtain the global optimum. Therefore, the model with the UCB acquisition function is selected and used to predict the optimal experimental conditions, as it is more robust and takes fewer steps to reach the global optimum.

4.1.3 Result of newly synthesized samples

The experimental conditions predicted by the model and their measured properties of the resulting new samples, namely, relative density after sintering and ionic conductivity, are listed in Table 4.2. All samples have a green density of approximately 62% so this property is not listed. Starting with 80 data points, 4 iterations (each predicting 5–6 samples) are performed to optimize the experimental conditions to obtain LATP with higher ionic conductivity. At the end of each iteration, the newly synthesized samples are fed back to the model for retraining. After the update, new experimental conditions are

Table 4.2: Recommended samples using BO and measured properties

		Experimental parameters			Measured properties	
Iteration	No.	rel. H_3PO_4 wt%	Temperature °C	Time min	rel. Density %	Ionic Conductivity S cm^{-1}
1	1	-7.5	1000	40	87.99	5.67×10^{-4}
	2	-7.5	1000	360	92.78	7.21×10^{-4}
	3	-7.5	1000	540	95.81	1.06×10^{-3}
	4	0.0	900	20	96.29	7.23×10^{-4}
	5	-7.5	1000	240	92.12	6.89×10^{-4}
2	6	-15.0	950	480	78.24	1.27×10^{-4}
	7	-22.5	950	540	80.00	4.93×10^{-5}
	8	7.5	950	540	75.22	6.67×10^{-5}
	9	0.0	850	30	92.56	4.91×10^{-4}
	10	0.0	1050	30	81.27	6.11×10^{-5}
3	11	0.0	900	40	95.80	4.07×10^{-4}
	12	-11.25	1000	480	97.47	9.06×10^{-4}
	13	-11.25	1000	510	96.11	7.87×10^{-4}
	14	-11.25	1000	450	97.52	1.09×10^{-3}
	15	-7.5	1000	450	95.00	4.02×10^{-4}
	16	-7.5	1000	510	97.36	4.37×10^{-4}
4	17	-15.0	1000	450	97.21	7.81×10^{-4}
	18	-11.25	1000	420	95.25	8.36×10^{-4}
	19	-15.0	1000	510	96.81	5.36×10^{-4}
	20	-15.0	1025	450	98.89	5.87×10^{-4}
	21	-11.25	1000	540	91.31	6.38×10^{-4}
	22	-11.25	1000	390	91.60	7.10×10^{-4}

predicted for the next iteration. This process forms a loop, which is repeated until the target number of iterations has been achieved. In total 22 new samples are synthesized.

It can be seen from Table 4.2 that the model quickly discovers a new sample (sample No. 3) with the second highest ionic conductivity ($1.06 \times 10^{-3}\,\mathrm{S\,cm^{-1}}$) in the first iteration. The ionic conductivity of this sample is very close to the highest one of all samples ($1.09 \times 10^{-3}\,\mathrm{S\,cm^{-1}}$), which shows a very good performance of the model. The comparison of experimental conditions shows that even though the sintering time of the new sample (540 min) is 60 minutes longer than that of the known maximum sample (480 min, with a deficiency of -7.25% H_3PO_4 at 900 °C), it can still maintain good performance. This indicates that LATP samples synthesized under this condition are stable against long holding time. On the other hand, comparing the densities of samples No. 2 (360 min) and No. 5 (240 min) it can be concluded that a shorter holding time is not enough for the samples to get fully sintered as the densities of these samples are less than those with a longer holding time. Therefore, the ionic conductivities of these samples are significantly lower than those of the fully sintered ones.

As BO involves a trade-off process between exploration and exploitation, it is important to explore unknown areas efficiently, which can help to find the global optimum in a scientific way. This can be well reflected in experiments of this study. It can be seen that in the first iteration, the model explores experimental conditions with holding time at 360 min (sample No. 2) and 240 min (sample No. 5), as the original experimental conditions of grid search have a large gap (uncertainty) of holding time between 60 and 480 minutes. Similarly, this can also explain why the model in the second iteration explores temperature intervals that have never been explored before, as these areas are subject to relatively large degrees of uncertainty. Though samples obtained in the second iteration show poor ionic conductivity (on average $1.59 \times 10^{-4}\,\mathrm{S\,cm^{-1}}$), it does not render this iteration a failure. This attempt is reasonable and can help the model to quickly explore this unknown region and excludes the possibility of an optimal value appearing in that region, which effectively reduces the number of tries it needs compared to the exhaustive method. Notably, starting from the third iteration, the range of experimental conditions that can be chosen has been expanded, more centrally, the extent to which the amount of precursor H_3PO_4 can be adjusted is increased, as its preparation can take a long time. It can be clearly seen that the model also undergoes a competitive process between exploration and exploitation in the third and fourth iterations. It first makes predictions (samples No. 12–14) with experimental conditions with a deficiency of -11.25% H_3PO_4, which has never been explored before (exploration). At this point, the model finds a new sample (sample No. 14) with a value of ionic conductivity which is as large as the previous maximum ($1.09 \times 10^{-3}\,\mathrm{S\,cm^{-1}}$). Then the model begins to make the most of this information and makes several attempts (samples No. 18, 21, 22) around this point (exploitation). It can be seen that the properties (ionic conductivity) of later samples are inferior to that of sample No. 14, indicating that the model has a good predictive performance. As a result, it can find the optimal value efficiently and quickly, reducing the number of required experiments.

The values of ionic conductivity (black line) during the four iterations and the values of the maximum (grey line) are plotted in Figure 4.3. It can be noted that the overall result shows a step-wise upward trend, illustrating the improvement of new samples during iterations. During the iteration, the model goes through a process of exploration and exploitation, which can be reflected in the fluctuating experimental results of ionic conductivity. The model first selects a sample that yields moderate performance ($5.67 \times 10^{-4}\,\mathrm{S\,cm^{-1}}$). Afterwards, the performance of samples increases with iterations and it quickly finds the second highest maximum (sample No. 3). From sample No. 11, the search space is extended (marked as the vertical grey dashed line in the middle) and the model quickly finds another global maximum (sample No. 14, marked as a red star). Overall, the results of the predictions prove that the model has a good ability to help researchers find another sample with maximum ionic conductivity under different experimental conditions where it has never been explored before. Using the Bayesian optimization method can help the experimentalist to quickly narrow the search space and hence can reduce the number of required experiments effectively.

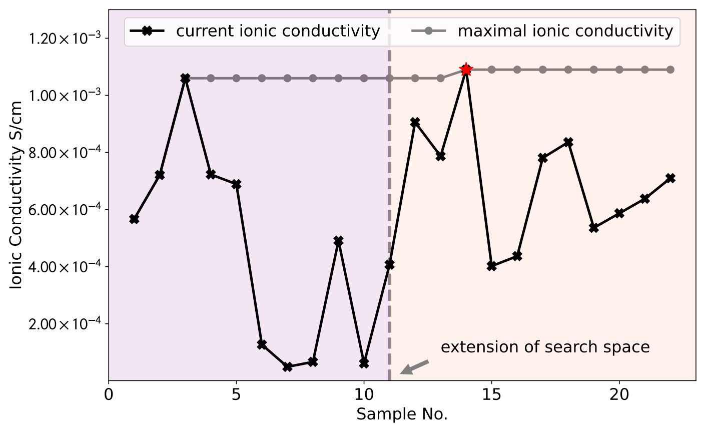


Figure 4.3: The values of ionic conductivity for each sample (black line) during the four iterations and the values of the maximum (grey line). The results of experiments show a fluctuating trend, reflecting the exploration vs. exploitation in the optimization process. The overall result exhibits a step-wise upward trend and it can be seen that the model has found the largest values of ionic conductivity (marked as a red star) within several iterations. Reproduced with permission [475].

To further explore why these samples (e.g., sample No. 14) have better performances than others, characterization measurements are performed to investigate the mechanisms governing the high ionic conductivity. Details are given in the next section.

4.1.4 Characterization of samples

A standard stoichiometric LATP sample reaches the highest ionic conductivity at a sintering temperature of 900 °C and a holding time of 30 minutes. With the aid of machine learning algorithms, the best properties are achieved for sample No. 14, which is sintered from a LATP batch synthesized with a deficit of −11.25 wt% in phosphoric acid and sintered at 1000 °C for 450 minutes. For a better understanding of why this sample has also reached ionic conductivity performance in the order of $1 \times 10^{-3}\,\mathrm{S\,cm^{-1}}$, even though synthesis and sintering conditions deviated from the standard procedure, the microstructure is analyzed. Figure 4.4 shows a comparison of the X-ray diffraction patterns of the standard stoichiometric LATP (0.0 wt%) sample and another LATP (−11.25 wt%) sample. While the stoichiometric sample is still phase pure in the expected NZP-structure after sintering, clear foreign peaks can be seen in the LATP (−11.25 wt%) sample. Among the foreign phases that have formed in addition to the NZP-structure, TiO_2 could be identified as a second phase.

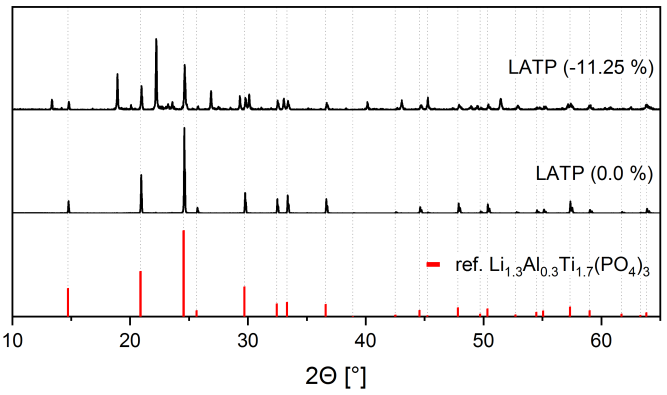


Figure 4.4: X-ray diffraction patterns for the LATP (−11.25 wt%) sample sintered at 1000 ℃ *for* 450 *minutes and one LATP (*0.0 *wt%) sample sintered at* 900 ℃ *for* 30 *minutes. A standard X-ray diffraction pattern of* $Li_{1.3}Al_{0.3}Ti_{1.7}(PO_4)_3$ *from the database is shown in red color for reference. Reproduced with permission [475].*

In addition, the different microstructure developments are shown in Figure 4.5. Figure 4.5a shows the LATP (0.0 wt%) sample sintered at 900 °C with a holding time of 30 minutes. There, the ionic conductivity in the order of $1 \times 10^{-3}\,\mathrm{S\,cm^{-1}}$ is achieved by a homogeneous and dense microstructure with small and uniform grains as well as very small and finely distributed pores. Despite the significantly different sintering parameters, Figure 4.5b shows a similar homogeneous microstructure with only slightly larger grains and pores for the LATP (−11.25 wt%) sample. The homogeneous grain size and dense structure are not typical for LATP sintered at this high temperature as the comparison in Figure 4.5c with LATP (0.0 wt%) sintered with these parameters shows. There, abnormal grain growth and the related micro-cracks in these large grains, due to a high thermal expansion anisotropy between a and c lattice parameters, shatter the microstructure and cause a drastic decrease in ionic conductivity [495, 496, 497]. Grain growth seems to be suppressed for the LATP (−11.25 wt%) sample by second-phase particles, which are visible as homogeneously distributed bright dots throughout the structure at the triple points of the grain boundaries.

The evaluation of the corresponding XRD-pattern leads to the assumption that the second phase is TiO_2 and this is reinforced by the results of the Energy Dispersive X-ray (EDX) analysis of one of these particles shown in Figure 4.6. The line-scan shows a clear increase in the titanium intensity in the EDX measurement at the position of the bright particle. The interaction between second-phase particles and migrating grain boundaries is known as the Zener-type mechanism in ceramic materials and can reduce grain growth. This is likely to be the reason for the moderate grain growth of the LATP (−11.25 wt%) sample at these high sintering temperatures [498]. It allows for densification of the microstructure

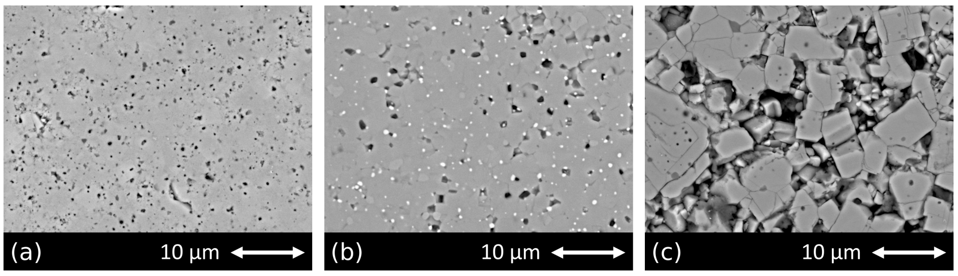


Figure 4.5: Microstructure of (a) LATP (0.0 wt%) sample sintered at 900 ℃ for 30 minutes; (b) LATP (−11.25 wt%) sample sintered at 1000 ℃ for 450 minutes and (c) LATP (0.0 wt%) sample sintered at 1000 ℃ for 450 minutes. Reproduced with permission [475].

that in turn results in an ionic conductivity in the order of $1 \times 10^{-3}\,\mathrm{S\,cm^{-1}}$. The above analysis serves to explain the possible mechanism why samples like No. 14 have better performances than others, which agrees well with the conclusions from previous study [486] where a deficiency of phosphoric acid in the synthesis can lead to the formation of $LiTiOPO_4$ and TiO_2. It seems that the second phases are the reason for the prevention of abnormal grain growth for sintering temperatures up to 1000 °C. Because the small grains are less susceptible to microcracking, a dense structure with high ionic conductivity is achievable even sintered at these high sintering temperatures. Experimental parameters for other samples, such as sample No. 3 ($1.06 \times 10^{-3}\,\mathrm{S\,cm^{-1}}$) synthesized with parameters (−7.5 wt%, 1000 °C, 540 min), are comparable to one of the samples (−7.5 wt%, 1000 °C, 480 min with ionic conductivity of $1.09 \times 10^{-3}\,\mathrm{S\,cm^{-1}}$) in the paper mentioned above and one can therefore assume that a comparable microstructure is the reason for the high ionic conductivity of this sample as well.

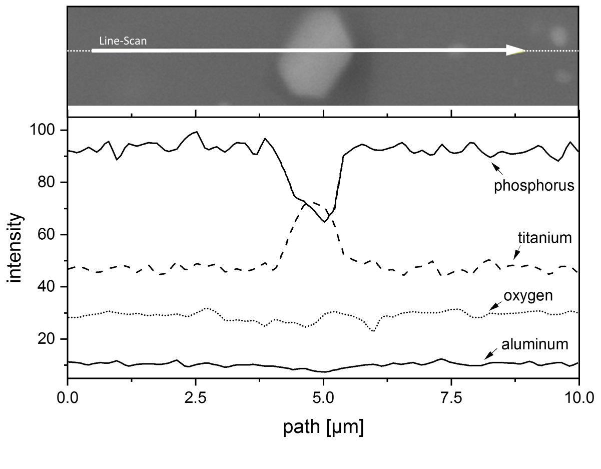


Figure 4.6: EDX-line-scan of a bright second phase particle in the structure of the sintered LATP (−11.25 wt%) sample. A significant increase can be clearly seen in titanium intensity at the position of the bright region, which indicates that the second phase is highly possible to be TiO_2. Reproduced with permission [475].

4.1.5 Conclusion

This study demonstrates that a data-driven materials design strategy based on Bayesian optimization using the Gaussian process regression model can be utilized to effectively design experimental conditions for synthesizing LATP, a promising candidate for solid electrolytes used in batteries. The design strategy can be divided into various sections. Firstly, virtual experiments are conducted to compare models that are built using different design strategies. The training data used in such experiments is collected from previous laboratory studies. It has been determined that the Upper Confidence Boundary (UCB) strategy results in the best performance. Next, the optimal model is chosen to predict new experimental parameters in order to produce a new sample with the highest achievable level of ionic conductivity. Following this, the corresponding characteristics of the newly sintered samples are analyzed and these results are utilized to improve the model in preparation for the subsequent iteration of the design process. The results show that within several iterations, newly synthesized samples guided by the model can achieve good performance with a maximum value of $1.09 \times 10^{-3}\,\mathrm{S\,cm^{-1}}$, which is in the same order of magnitude as the maximum Li-ion conductivity that LATP can achieve. In addition, the range of the search space can be dynamically adjusted during the experiment, making this method flexible according to the researcher's needs. Moreover, it can help the researcher to quickly explore the boundary of the range of experimental parameters that can yield samples with good performance, thus helping to design experiments in an effective and reasonable way to reduce the number of required experiments. It is noteworthy that the main focus of this work lies in the single objective optimization domain, wherein only ionic conductivity is considered. This entails simplifying the optimization problem, as other properties - such as sintered density - may also influence the property of interest. Therefore, taking these factors into account and treating the problem as a multi-objective optimization problem may further improve the performance of the model and thus lead to better samples. This issue deserves further investigation.

4.2 Microstructure analysis and parameter optimization for simulation

Porous membranes are extensively used in various fields due to their unique features and it is crucial to rigorously characterize their microstructures to understand corresponding properties and enhance their performance for specific applications. A promising method for quantitatively analyzing porous structures involves using physics-based generation of such structures at the pore level, validating them against real experimental microstructures, and using data-driven algorithms to build process-structure-property relationships. A Variational AutoEncoder (VAE) neural network model is utilized in this study to characterize the 3D structural information of porous materials and then represent them as low-dimensional latent variables. These variables can also used to model the structure-property relationship and tackle the inverse problem of process-structure linkage by implementing the Bayesian optimization method. The adopted methods provide an approach to automatically extract structural information through unsupervised learning, which can effectively characterize porous microstructures. These results have been published in [499] and the following sections are based on this publication. An overview of this study is shown in Figure 4.7.

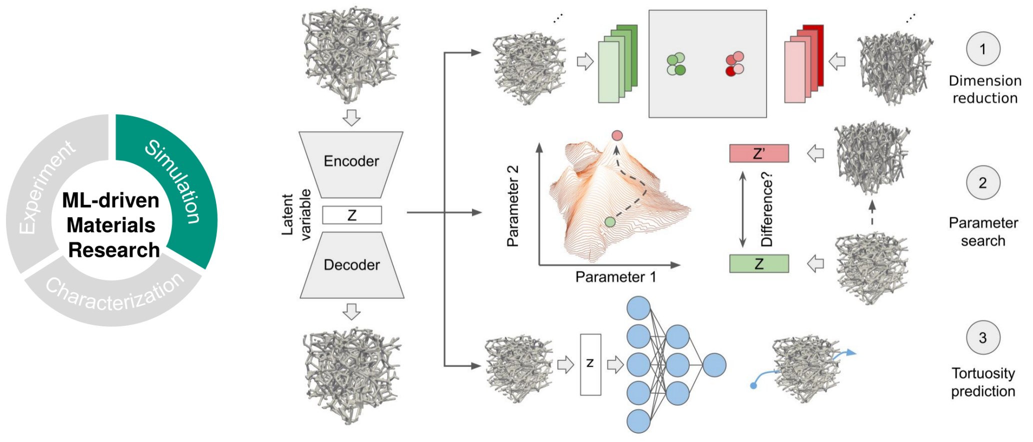


Figure 4.7: An overview of building a fast prediction model for linking process-structure-property relationships in porous microstructures with a neural network model. This neural network model can extract structural information from microstructures and represent them in low-dimensional variables (1). The trained model can be used to solve the inverse design problem in combination with other optimization strategies (2) and can also be used to predict the quantity that needs to be solved by simulation (3). Reproduced with permission [499].

4.2.1 Materials and data

In this section, 3D digital twins of polymeric porous media are introduced. The data sets representing the digital twins are obtained by applying a Voronoi-tessellation-based generation algorithm and by converting reconstructed computer tomography scans from the real membrane microstructure into 3D data of voxels [500].

Generated porous microstructure

3D polymeric porous microstructures are generated by an in-house implemented generator based on a previous study [500]. The generation process is performed by two main steps: (i) random points are dispersed inside the simulation volume for the construction of the Voronoi-diagram, which will divide the domain of interest in Voronoi regions (cells) according to the nearest neighbor rule. Coordinates of these points are determined using a deterministic working random function which can recreate the same random distribution of points by using the (random) seed parameter; (ii) porous microstructure is generated on the framework constructed by edges of the polyhedral Voronoi cells, with stanchions and pores being formed parametrically during the generation process. After these two steps, certain post-processing can be followed such as placing multiple spheres with possible varying radii by obeying a logarithmic distribution function and smoothing the structure by appropriate Gaussian filters. Porous membranes created using the algorithm described above can hold diverse features.

The most influencing factors that determine the morphology of porous microstructure are: the density (number) of Voronoi seed points, the prescribed thickness of ligaments, the resulting overall porosity as well as the pore size (directly related to the number of Voronoi points), and the stretching factor for the x-, y- and z-directions, which orients the ligaments in certain directions to introduce anisotropy to the microstructure. For stretched microstructures, the produced Voronoi diagram is manipulated according to the stretching factor followed by performing the microstructure generation. As a result, the stretching process only leads to a distortion of the Voronoi region while the primary features of the microstructure remain unaffected.

Features of generated microstructures can be controlled by varying parameters for digital generation, which can yield microstructures with a wide variety of morphology and properties, e.g., the pore size of the membrane structure is indirectly defined by the number of Voronoi points. A smaller number of Voronoi points leads to larger pores and thus also to a higher porosity. In this work, the most important parameters for the generated microstructures are selected as: (i) the number of Voronoi points n: 40–170; (ii) stretching factor e: 0.5–1.0 (only one direction at a time, smaller value stands for stronger stretch); (iii) stanchion radius r: 3.4. Choices of these parameters are made so to cover various morphologies and to match the algorithm-generated microstructures as closely as possible with the real experimental structures (detailed in the next section). For real membranes, their microstructural characteristics can be controlled by processing parameters during the production process, such as temperature, humidity, evaporation speed, and composition of the added solvents [501]. In this work, the research interest is how to use the Bayesian optimization algorithm to quickly find the parameters for generating microstructure of porous membranes with given morphology, however, this principle is promising to be adapted and transferred to real production scenarios. Since the parameters used for digital data generation do not directly translate into real processing parameters, further studies are required, e.g. a quantitative and qualitative analysis of their linkages and influences on resulting microstructures.

In this work, microstructures are characterized by using data-driven descriptors, i.e., features of microstructures are learned by the neural network model automatically in an unsupervised manner and microstructures are represented as low-dimensional latent variables. The analysis of latent variables and their association with microstructural properties are discussed in the results sections. For the training of the Variational AutoEncoder (VAE) neural network model used for data-driven characterization, a data set which consists of a total of 10000 porous microstructures (generated by uniform sampling within the selected range of parameters for generation) with a domain size of 200^3 cells (about $30 \times 30 \times 30$ µm^3 corresponding to the real physical world) is produced by algorithm-based generation (a few representative ones are depicted in Figure 4.8a–d). These microstructures serve as the inputs for training the model. For generating digital membranes, the in-house developed PACE3D simulations tool was used and it took about 15 seconds (averaged on 100 generations on a platform with CPU Intel Xeon Gold 6146 $\times$ 2, 24 cores, 48 threads) to generate a single sample (and in total about 42 hours for 10000 samples as the data set). To meet the input requirements of the proposed VAE model, generated microstructures are resized to 128^3 using spline interpolation [502]. The morphology of the resulting produced porous microstructure is controlled by processing parameters during generation. In addition to the generated microstructures which are stretched in a single direction, another batch of microstructures (3000) which are stretched in two directions (a few representative ones are depicted in Figure 4.8e–g) are also generated to serve as the additional test data set for validating the VAE model's ability to characterize porous microstructures.

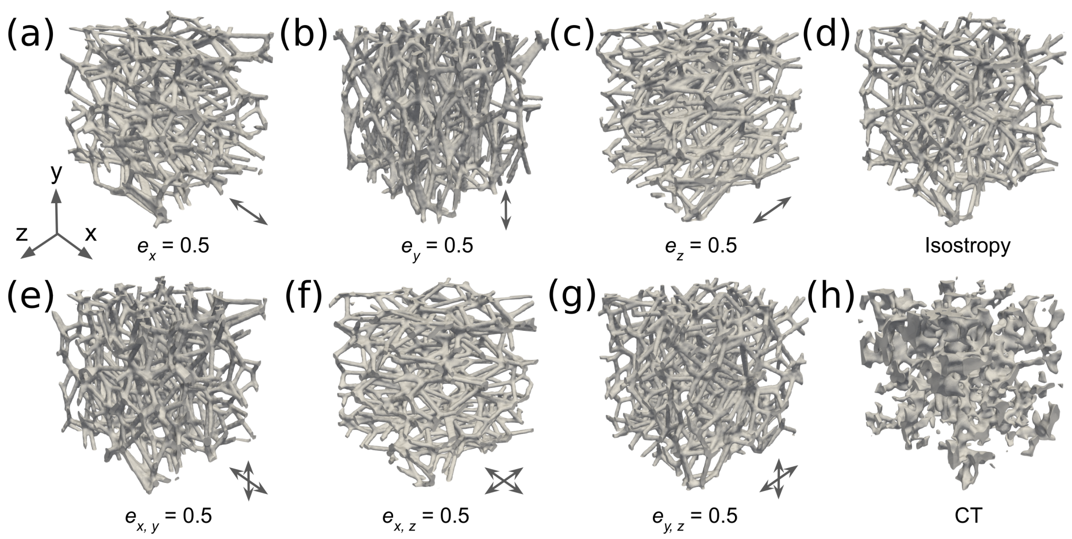


Figure 4.8: Examples of generated porous microstructure stretched in different directions. Samples (a)–(c) are stretched (with 0.5 *stretching intensity) only in x-, y-, and z-directions, respectively, while the sample in Figure (d) is an isotropic one (no stretch). For samples (e)–(g) they are stretched in two directions at the same time. Sample (h) is a real reconstructed structure from the experiment. Reproduced with permission [499].*

Real experimental porous microstructure

The geometric anisotropy is one of the most interested properties, as this can reflect many microstructural properties of the porous microstructure, such as the permeability in different flow directions. Hence, quantification of the degree of average anisotropy in the real porous microstructure is one of the main interests. In this study, real 3D images of three different microstructure samples are obtained by high-resolution X-ray Computer Tomography (XCT), whose anisotropy is of the most interest. The XCT scans are carried out with ID16B-NA beamline at the European Synchrotron Radiation Facility (ESRF) in Grenoble [503]. The beamline has a minimum and maximum beam size of 50.0×50.0 nm^2 up to 1.0×1.0 µm^2 using an energy range of 6.0 keV to 65.0 keV. The reconstructed 3D microstructure samples are composed of 900 equally spaced 2D images with the resolution of 150 nm pixel^{-1}. The size of the resulting segment of the real membranes is $900 \times 500 \times 500$ voxels (approximately $75 \times 135 \times 75$ µm^3) for each sample. The resulting 3D data are obtained as a stack of 8-bit (256 intensity levels) gray-scale images in XCT imaging, which is further required to be filtered and then segmented to obtain a clear separation between the microstructure and the pore space. To reduce the noise in the images, a Gaussian 3D filter with variance $\sigma^2 = 3$ voxels is applied first. Then the threshold for segmenting the images into zeros and ones (representing the pore space and the membrane material, respectively) is continuously adjusted until the experimentally obtained porosity values match up with the porosity of the segmented microstructures [504]. To meet the need for the input of the model, the original large microstructures are cropped into smaller cubes with a size of 200^3 and then are resized to 128^3. Among them, 15 structures are randomly selected to validate the effectiveness of the neural network model to extract microstructural features and to relate them to the interested properties (porosity and anisotropy). These properties are estimated by using the clustering method (finding the closest neighbor) in the low-dimensional space, to be more specific, their properties are inferred from the most similar generated microstructure found in the training data set. More details about these real porous membranes are described in previous work [505]. Figure 4.8h displays one of the real reconstructed experimental 3D microstructures.

4.2.2 Low-dimensional representation

The architecture of the VAE model used in this work is summarized in Table 4.3. This architecture is obtained empirically by trial and error based on the performance of validation data (10% of the whole data set). An exhaustive systematic study of varying the architecture was not performed, which may be helpful for improving the performance. For the inputs, as the original data is already in binary format (pore: 0, structure: 1), no scaling is performed. Binary cross-entropy is chosen as the loss function for measuring the reconstruction difference. Besides, Adam optimizer (learning rate: 10^{-5}, moment terms β_1: 0.9, β_2: 0.999) is used as the optimization algorithm for stochastic gradient descent to train the neural network model with batch size of 64 for 1000 epochs. The models were trained on a single NVIDIA A100-80GB GPU and it took about 14 hours to finish the training process (averaged on 10 trials of training).

Table 4.3: Architecture of the VAE model.

Model	Functions	Parameters	Model dimensions
Encoder	Input	-	128, 128, 128, 1
	Conv3d + ReLU	filters = 16, kernel = 4, stride = 2	64, 64, 64, 16
	Conv3d + ReLU	filters = 32, kernel = 4, stride = 2	32, 32, 32, 32
	Conv3d + ReLU	filters = 64, kernel = 4, stride = 2	16, 16, 16, 64
	Conv3d + ReLU	filters = 128, kernel = 4, stride = 2	8, 8, 8, 128
	Flatten	-	65536, 1
	Dense + ReLU	2048	2048, 1
	Dense × 2	256 × 2	μ: 256, 1; σ: 256, 1
Decoder	Dense + ReLU	65536	65536, 1
	Reshape	-	8, 8, 8, 128
	ConvTranspose3d + ReLU	filters = 64, kernel = 4, stride = 2	16, 16, 16, 64
	ConvTranspose3d + ReLU	filters = 32, kernel = 4, stride = 2	32, 32, 32, 32
	ConvTranspose3d + ReLU	filters = 16, kernel = 4, stride = 2	64, 64, 64, 16
	ConvTranspose3d + Sigmoid	filters = 1, kernel = 4, stride = 2	128, 128, 128, 1

A total number of 10000 porous structures are generated using the physics-based algorithm for evaluating the VAE model. This data set is divided into 80% of the training part, 10% validation part, and 10% test part. The training part is used to train the model and the validation part is used to select good hyper-parameters for the model and the training algorithm. After identifying the satisfying architecture of the model, the training and validation parts are merged as a larger training data set to retrain the model for further improving its performance. The remaining 10% test part is utilized for evaluating the final performance of the model. Figure 4.9 shows the total loss of the VAE model during the training process of 1000 epochs. In the beginning, the loss decreases quickly until this changing rate eventually drops significantly after about 150 epochs. Then, the loss of the test set declines slowly and gradually converges and finally levels off in the last 200 epochs, while the loss of the training set is still slowly continuing to decrease. A gap between the loss of the training set (blue solid line) and the test set (grey dashed line) can be observed after about 250 epochs of the training, and it is speculated that the model appears to be slightly over-fitted.

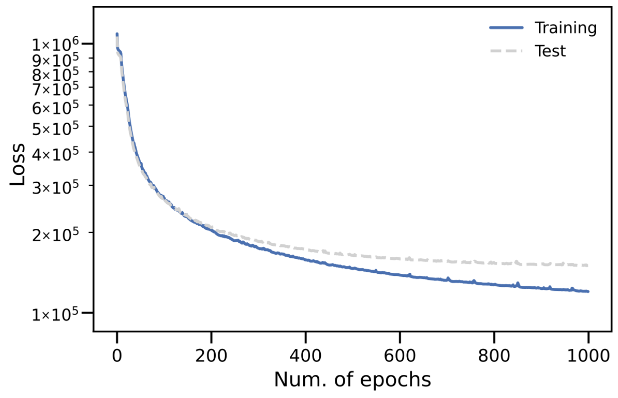


Figure 4.9: Total loss of the neural network model during the training process. The loss of the training set is marked with a blue solid line while the loss of the test set is marked with grey dashed line. Reproduced with permission [499].

The encoder part of the VAE model processes the algorithm-generated structures and thus represents them as low-dimensional latent variables. It is found that in this latent space, the most interested structural information is scattered across multiple dimensions and no single dimension is found to be obviously well disentangled (detailed in Appendix A). Hence, to further refine the information and to visualize the result, the Principal Component Analysis (PCA) is performed on these latent variables. The distribution of the generated porous microstructures in the space of the first three principal components is shown in Figure 4.10. The top two subplots (a and b) exhibit the space distribution of the training set (points without edges) and test set (points with black edges). The magnitude of porosity is reflected by the color shade of the data points (the darker the color, the larger the value of porosity) and it can be found that the value of porosity gradually increases along the negative direction of the third principal component (PC3) axis (right to left). Hence, it is concluded that PC3 mainly captures the porosity information of microstructures. Besides, it can also be observed that the number of Voronoi points used in the microstructure generation process decreases (indicated by the sizes of the data points) along the negative direction PC3 axis. This is in accordance with the properties of the generated microstructure as more skeletal structures are produced with increasing Voronoi points during the generation process, which will take up more space and hence result in a smaller porosity value. In addition, the subplot on the top of Figure 4.10b shows the distribution of stretching directions along the PC3 axis and it is observed that all the microstructures are uniformly distributed along the PC3 axis. Therefore, PC3 is inferred to contain less amount of directional information and only capture information related to porosity.

The distribution of the generated microstructures in PC1 and PC2 is presented in Figure 4.10a and it can be inferred that the directional features of the microstructures correlate mainly to these two principal components. In this figure, the microstructures are color-coded according to their stretching directions. It can be seen that microstructures without the preferred stretching direction are concentrated in the center, while structures with different stretching directions are distributed along their respective orientation branches and lie at an angle of about 120 degrees to each other. The embedded subplot depicts the stretching strength of the microstructure in the y-direction, with lighter colors implying a stronger degree of stretching. It can be observed that the color changes from darker to lighter from the center outward in the corresponding branch (microstructures stretched in y-direction) and the farther the microstructure is from the center, the stronger the stretching strength is. Besides, the top and right subplots (the density distribution of the stretching direction) also demonstrate that PC1 and PC2 together can clearly capture the directional information of microstructures. Furthermore, comparing the distribution of the training set (points without edges) and validation set (points with black edges), the model performs well on both and can accurately characterize their stretching intensities and directions.

To further demonstrate the model's ability to extract microstructural information, an additional test data set which contains microstructures stretched in two directions is generated. These new microstructures are also processed by the VAE model to extract structural information, represented as low-dimensional variables and further projected to principal component space. Similar results are shown in Figure 4.10c–d. From Figure 4.10d (PC2 vs. PC3) it is known that PC3 can also capture the information related to porosity (and the associated number of Voronoi points). Along the negative direction of the PC3 axis, the color of the data points changes continuously from light to dark, implying the increase of porosity values of microstructures, and it corresponds to a decrease in the number of Voronoi points set in the microstructure generation process. This finding is consistent with the conclusions from the previous analysis based on the microstructures stretched in a single direction. Figure 4.10c (PC1 vs. PC2) shows the distribution of stretching directions of all generated microstructures. From the figure, it can be known that the model shows good performance in extracting the directional information of new microstructures that have not shown in the training period and can project them to reasonable positions according to their stretching direction. For example, the model projects microstructures stretched in both x-and y-directions (purple) to the place between microstructures stretched in only a single direction (x or y, blue or orange, respectively). This further demonstrates the model's robust ability to extract microstructural information and shows its robust performance on new unseen data.

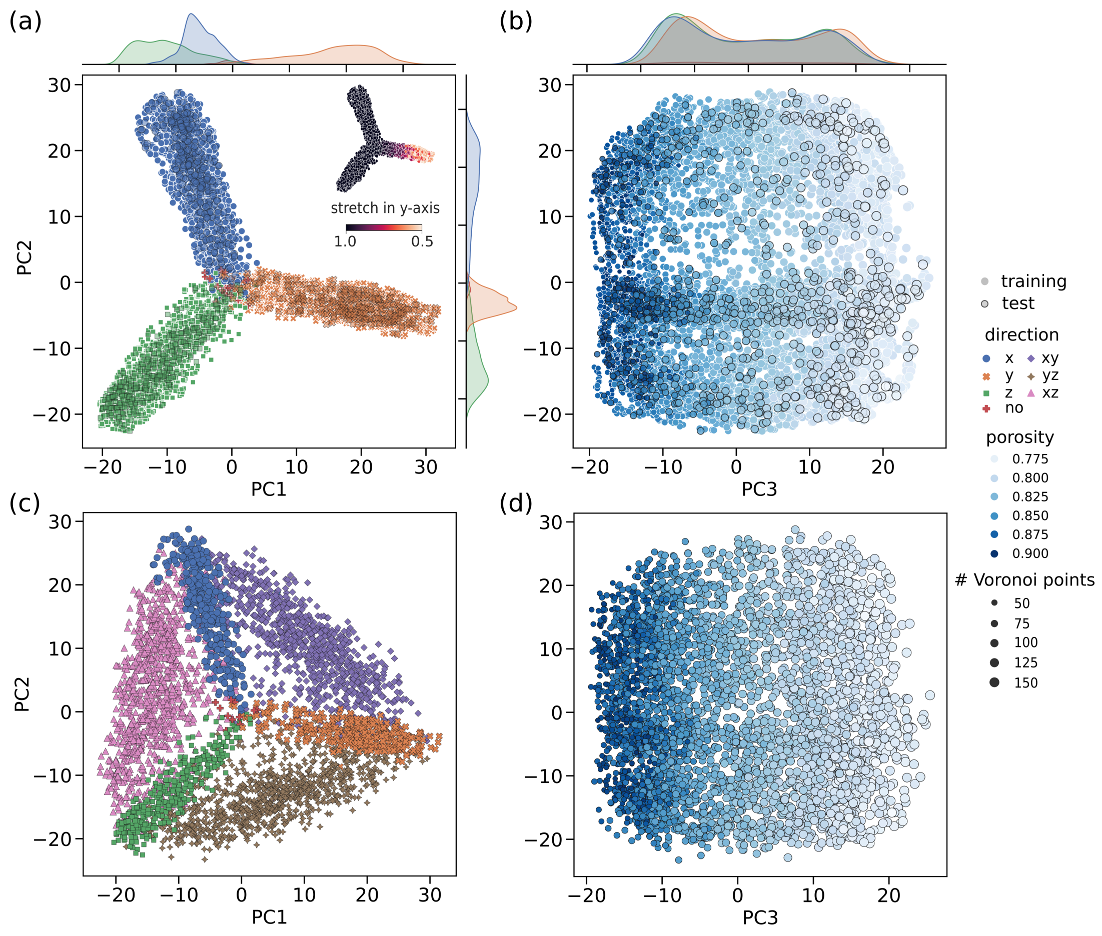


Figure 4.10: The distribution of porous microstructures in the first three principal component spaces (PC1–3). (a) and (b) exhibit microstructures with a single stretching direction while (c) and (d) show microstructures having at most two stretching directions. It is found that the first two principal components (PC1–2) together capture the information of stretching direction and strength while PC3 mainly contains information related to porosity/Voronoi points of microstructures. Reproduced with permission [499].

4.2.3 Characterization of experimental microstructure

The VAE model is further applied to the 3D experimental microstructures which are reconstructed from CT images. The above analysis of the model's performance on generated microstructures proves that the trained VAE model shows robust performance and only slight overfitting is observed. As before, the trained neural network model is used to extract microstructural features from the reconstructed experimental microstructures and to represent them as low-dimensional latent variables. Together with the generated microstructures, PCA is performed on these latent variables. In the principal component space, microstructures with similar properties will be clustered closer to each other and it is intended to analyze the interested properties of the experimental microstructure according to the meanings represented by PC1–3. These results are shown in Figure 4.11.

First, the porosity value of the experimental microstructure is estimated according to PC3, which correlates most with the information about porosity. From Figure 4.11b it can be seen that the reconstructed experimental microstructures are concentrated in the central place of all the data points, locating at the position in PC1 axis where it should correspond to a porosity value of about 0.84. The porosity value of experimental microstructures can be estimated by finding the closest algorithm-generated microstructures (i.e., nearest neighbor) where the "similarity" is measured by calculating the difference in PC3. The calculated porosity of reconstructed experimental microstructures, as well as the estimated (predicted) porosity value according to the closest found algorithm-generated microstructures, are listed in Table 4.4. It is found that there are some differences between the numerically calculated porosity value and the estimated value. By comparing the reconstructed experimental and algorithm-generated microstructures, it is speculated that such kind of discrepancy is mainly caused by some dissimilarities between real reconstructed experimental and algorithm-generated microstructures. Though our generated microstructures can resemble the morphology of real microstructures, artifacts caused by e.g., noise during the reconstruction process of experimental microstructures, can lead to some deviations. As a result, this impairs the performance of the model's ability to characterize microstructures to a certain extent.

In addition, the limited amount of the data used to train the model somehow restricts the model's ability: only about 10000 microstructures are adopted and they may not be able to completely cover all possible configurations and morphologies of the real experimental microstructures. It is therefore understandable that for some microstructures (for example, No. 2 and No. 13, whose porosity values are located at the marginal place of the porosity distribution of the training set), no good corresponding algorithm-generated microstructures are identified to estimate the porosity accurately.

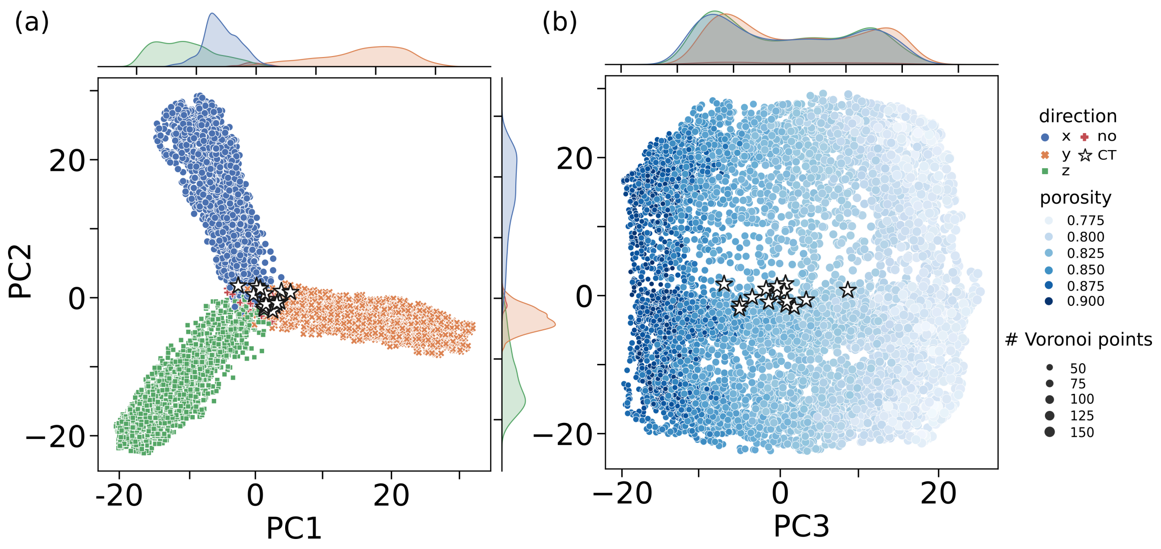


Figure 4.11: The distribution of algorithm-generated porous microstructures and real reconstructed microstructures from experiments (marked as CT). In (a) (PC1 vs. PC2) the reconstructed experimental microstructures are concentrated near the center and biased in the y-axis. Their microstructural anisotropy (measured in stretching direction) is estimated by finding the closest algorithm-generated microstructures in PC1–2. The porosity value of reconstructed experimental microstructures is estimated in PC3 (b) as it is found to correlate with the information about porosity. Reproduced with permission [499].

In addition, it is possible to estimate the stretching direction of reconstructed experimental microstructures in PC1–2 as they represent directional information of microstructures. From Figure 4.11b it can be clearly seen that experimental microstructures (marked with white stars) are concentrated near the center and biased in the y-axis. Similarly, the distance (similarity) can be calculated in PC1 and PC2 to find the closest algorithm-generated microstructures to estimate the stretching direction of the reconstructed experimental microstructures. Besides, the anisotropy of the microstructure can be judged accordingly. In the present work, the tortuosity is numerically calculated by the simulation method, which is used to approximate the anisotropy of the microstructure, as it can indirectly represent the directional orientation of the microstructure. More specifically, if a microstructure is stretched in a certain direction, the tortuosity value should be small, which means that the microstructure shows a high anisotropy in this direction. The calculated tortuosity values in x-, y-, and z-directions of the reconstructed experimental structures are regarded as their true values and they are listed in Table 4.4. It can be seen that all reconstructed experimental microstructures have the smallest values in the y-direction, implying that they all tend to have a relatively large stretch along the y-direction. It is worth noting that tortuosity values in each direction do not differ much, which indicates the reconstructed experimental structures show only weak anisotropy. The average of the ratio of tortuosity in the y-direction to the x- and z-directions, respectively, is used to approximate the degree of anisotropy in the y-direction. The estimated stretching intensities of the reconstructed experimental structures according to their closest generated microstructures are given in Table 4.4. It is known that for most of the real experimental microstructures, the estimated stretching directions and intensity values are satisfyingly accurate. However, two wrong estimations of the stretching direction for samples No. 4 and No. 6 are made. This discrepancy is understandable, as the reconstructed experimental microstructures possess only very weak anisotropy (slightly stretched) so inherently it is hard to infer accurately. Such phenomena can also be observed in Figure 4.11a (PC1 vs. PC2) where the microstructures tend to cluster close to the center, and a slight deviation may lead to misjudging the stretching direction. In general, the VAE model performs well in extracting directional information of the reconstructed experimental microstructure, and their stretching direction (anisotropy), as well as stretching strength, can be estimated with high accuracy.

Table 4.4: Numerically calculated property values of experimental microstructures and their estimated (predicted) values.

	Porosity			**Tortuosity**			**Geometric anisotropy**				
Sample No.	**C**	**P**	**Diff. %**	**Axis** x	y	z	**Direction** C	P	**Intensity** C	P	**Diff. %**
1	0.846	0.822	-2.791	1.162	1.121	1.161	y	y	0.965	0.980	-1.554
2	0.894	0.836	-6.441	1.101	1.077	1.105	y	y	0.976	0.960	-1.639
3	0.842	0.839	-0.291	1.165	1.119	1.162	y	y	0.962	0.980	1.871
4	0.866	0.858	-0.862	1.127	1.104	1.134	y	**x**	0.977	0.980	0.307
5	0.850	0.825	-2.919	1.162	1.111	1.159	y	y	0.958	0.960	0.208
6	0.857	0.824	-3.740	1.143	1.125	1.137	y	**no**	0.987	1.000	1.317
7	0.887	0.835	-5.755	1.106	1.086	1.115	y	y	0.978	0.980	0.204
8	0.861	0.833	-3.207	1.140	1.109	1.139	y	y	0.973	0.990	1.747
9	0.827	0.829	0.242	1.184	1.147	1.188	y	y	0.967	0.950	-1.758
10	0.845	0.839	-0.702	1.167	1.130	1.157	y	y	0.972	0.970	-0.205
11	0.790	0.806	2.046	1.227	1.192	1.250	y	y	0.962	0.950	-1.247
12	0.836	0.829	-0.834	1.169	1.122	1.184	y	y	0.953	0.980	2.833
13	0.898	0.827	-7.869	1.096	1.073	1.105	y	y	0.976	0.980	0.409
14	0.844	0.830	-1.641	1.156	1.115	1.173	y	y	0.957	0.950	-0.731
15	0.842	0.825	-1.967	1.169	1.140	1.158	y	y	0.980	0.970	-1.020

C: calculated value, P: predicted value. Geometric anisotropy is measured in stretching strength.

4.2.4 Inverse problem of the process-structure linkage

The VAE model is used to extract structural information and represent microstructures as latent variables. With these low-dimensional variables, mathematical metrics such as L_p-norm, can be calculated to measure the difference between various microstructures. Taking advantage of this kind of data-driven descriptors of microstructures, an inverse design problem of the process-structure linkage is explored in this section. In order to verify the validity of the model's performance on unseen microstructures, four microstructures (marked as A, B, C, and D) with morphologies that are quite different from the training data are intentionally generated and their corresponding true processing generation parameters are listed in Table 4.5. The corresponding generation parameters to produce these microstructures are assumed unknown to us and the goal is to find these values. Such a problem is treated as an optimization task, and an active learning approach based on Bayesian Optimization (BO) is used to solve it effectively.

Table 4.5: Results of Bayesian optimization in finding parameters for digital generation of target microstructures.

Sample No.		**Stanchion radius**	**Number of Voronoi points**	**Stretch**			**Difference in latent variable space**
				x	y	z	
A	Real	3.40	100	1.00	0.75	1.00	-
	Median	3.58	100	1.00	0.75	1.00	3.82
	Best	3.39	100	1.00	0.75	1.00	0.18
B	Real	3.40	180	1.00	0.75	1.00	-
	Median	3.57	183	1.00	0.75	1.00	4.14
	Best	3.28	179	1.00	0.74	1.00	1.51
C	Real	5.40	180	1.00	0.75	1.00	-
	Median	5.35	184	1.00	0.75	1.00	8.03
	Best	5.57	179	1.00	0.75	1.00	2.02
D	Real	3.40	180	0.65	0.55	1.00	-
	Median	3.77	179	0.61	0.50	1.00	8.23
	Best	3.40	178	0.63	0.55	1.00	3.12

The convergence plot of the difference between the target microstructure and generated microstructure being minimized during the optimization process is shown in Figure 4.12. At the beginning of the optimization, 20 random microstructures are generated to serve as the input to initialize the Bayesian optimizer, and the number of iterations is relaxed to 1500 to ensure that the optimal parameters can be reached as close as possible. For each target microstructure, the optimization process is repeated 10 times to draw basic statistical information for better evaluating the optimization results. Among the results (marked in grey lines), the round of iteration result corresponding to the median is marked in blue for better visualization. To visually display the optimization process, above each subplot, the corresponding structures generated by the found processing parameters at the initial state, at one-half and one-fourth states of initial difference values, and the final found value, are also drawn respectively. True processing parameters that generate the structures and results of the Bayesian optimization are summarized in Table 4.5.

From the BO results it is found that the corresponding processing generation parameters for the target microstructures can be satisfyingly estimated despite a certain amount of discrepancy. Besides, it can be observed from Figure 4.12 that during the optimization process, the similarity of the generated microstructure to the corresponding target is improved significantly at the very beginning part of the optimization process and the degree of improvement gradually slows down as the number of iterations grows. It is noticeable that the optimization process has also experienced several plateaus, which is due to the nature of BO as it takes both the mean estimation and the uncertainty in the meta-model into consideration (the so-called exploration vs. exploitation). Comparing the optimization results of the four target microstructures, it is found that for microstructures (A, B in Table 4.5) that are more similar to the original training set, the VAE model can better characterize them and hence can better distinguish them, resulting in fewer iterations needed to find the optimum. On the other hand, for those microstructures (C, D in Table 4.5) that are more different from the original training set, the model is also able to extract their structural information and differentiate between them, though the performance is slightly poorer. Thus, more iterations are required in these cases for BO to converge to the optimal value. Apart from that, the variance of these results is also larger than those of microstructures A and B, indicating that the optimization process is less stable. Overall, the process of combining data-driven microstructural descriptors learned by the VAE model with the Bayesian optimization can provide a satisfying solution to the inverse problem of structure-process linkage, i.e., finding the processing generation parameters of unknown target microstructure. This further validates the ability of the VAE model to characterize the porous microstructures and proves the applicability of the Bayesian optimization method in solving the considered problem effectively. Besides, finding the appropriate parameters that can generate microstructure resembling real experimental microstructure will be an interesting extension of this strategy and is of interest for future exploration.

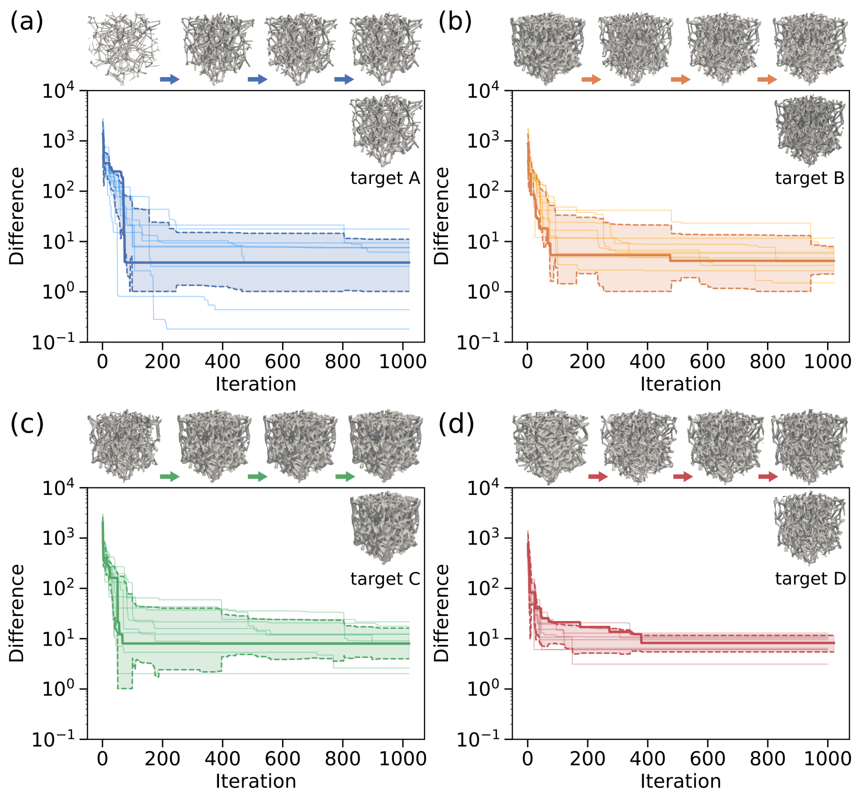


Figure 4.12: Iteration history of the Bayesian optimization in solving the inverse problem of process-structure linkage for four different target microstructures (a–d). For each target microstructure (A–D), the optimization process is repeated 10 *times and the result corresponding to the median of final differences is highlighted with a blue line. The difference between the target microstructure and the generated one is measured by calculating the Euclidean distance in the latent space. Reproduced with permission [499].*

4.2.5 Structure-property linkage

Transfer learning for predicting tortuosity

The performance of the neural network model is often related to the setting of the initial weights of the model. Hence, good choices in setting these initial weights play an important role in obtaining a good neural network model. The VAE model was trained on generated porous microstructures in an unsupervised manner and the model has learned some knowledge about extracting the structural information. Hence, it is expected that re-using parts of the trained model can show better performance on prediction tasks such as building the structure-property linkage. To be specific, the encoder part of the VAE model is taken out to serve as the basic part of building a neural network-based regressor, that is, these pre-trained weights are re-used and are believed to be able to yield better results than random initialization of the weights, especially when only limited amount of labeled data is provided. This idea is one of the key points in Transfer Learning (TL) [506].

In this work, a structure-property linkage is built by training another neural network model, that is, the relationship between porous microstructures and their corresponding tortuosity values (described in Appendix B). For this purpose, an additional 300 new unseen structures (250/50 training/test split) with large differences (number of Voronoi points n = 180–250, stanchion radius $r = 2.8$) from the training set in previous sections are generated and simulations are performed to calculate tortuosity values (labels). Fine-tuning is adopted as a means to achieve transfer learning. More specifically, the whole process is divided into three steps: (i) transferring architecture and weights of the encoder: the pre-trained encoder part of the VAE model is taken out to serve as the microstructural information extractor and it is the basic building block for the new neural network model. It should be noted that no sampling process is performed on the latent variables (unlike that during the previous training of the VAE model); (ii) adding dense layers at the end of the encoder: immediately following the output of the encoder, two more dense layers (64 ReLU) are connected, and a final linear output neuron is added to yield the predicted scalar value (tortuosity). The weights of these layers are randomly initialized using the Xavier uniform initialization method [507]. Furthermore, a weak L_2 constraint (10^{-5}) is imposed as a regularization term to mitigate the overfitting phenomenon; (iii) fine-tuning weights of the model: the Mean Squared Error (MSE) is chosen as the loss function for this regression task. Besides, the Adam optimizer [508] is adopted as the optimization algorithm for the training and the parameters are set to be the same as those used in training the VAE model. The optimization is set to last for 100 epochs to ensure convergence. During the training process, the encoder part is frozen so only the weights of fully-connected layers are updated. It is noted that for the sake of brevity, other network architectures and training strategies are not exhaustively investigated in this study.

To demonstrate the benefit of employing the TL technique for developing the structure–property linkage model, another neural network model, which is built with exactly the same architecture but with all weights initialized randomly (rather than using the pre-trained weights of the encoder), is also trained for comparison (referred to as the "training from scratch"). The mean-squared errors are calculated on both the training and testing data to evaluate the performance of these two models trained with different strategies. To better assess the models' performance, the training process and evaluation are repeated 20 times and the results are shown in Figure 4.13. It can be noticed that results obtained from the TL method (averaged MSE: 0.00401 on training data and 0.00502 on test data) are better than those from the "training from scratch" method (averaged MSE: 0.00460 on training data and 0.00567 on test data). Besides, results from the TL method are also significantly more stable as they exhibit smaller variance. For both models, the errors on the test set are slightly larger than those on the training set, but the difference between the TL strategy is smaller than that with the "training from scratch" method, indicating that models obtained with the transfer learning strategy are more robust in extrapolation. It is speculated that such a privilege may come from the anti-overfitting effect introduced by freezing the weights in the pre-trained encoder model. Furthermore, although occasionally the model trained from scratch can achieve a smaller value of MSE on the training set, its performance on the test set is apparently worse than that of the TL strategy. Based on these results, it is inferred that the transfer learning strategy can facilitate the development of the structure-property prediction model in many aspects such as improving its accuracy and stability, especially when only a limited amount of data is provided. These results are consistent with the previous hypothesis that the model trained in an unsupervised manner has gained useful knowledge from the data and can be transferred to other downstream tasks which helps achieve better performance. Moreover, it should also be mentioned that there are many other model architecture designs and training strategies. Their impacts on the results of transfer learning, and how useful transfer learning can be in different application scenarios, are worth exploring and are of interest for future work.

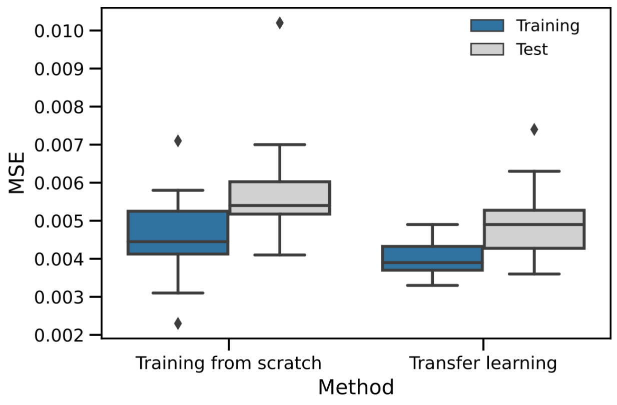


Figure 4.13: Mean Square Errors (MSE) of predicted tortuosity on both the training set and test set for two neural network models with the same architecture but trained with different strategies: one is trained from scratch and another adopts the transfer learning strategy. Reproduced with permission [499].

4.2.6 Conclusion

The Variational AutoEncoder (VAE) neural network model is used to characterize the three-dimensional microstructures of porous membranes. The model is used to extract structural information and represent it with low-dimensional latent variables as data-driven descriptors. The ability of the model to characterize porous structural information is verified in three aspects regarding the establishment of process–structure–property linkages. First, Principal Component Analysis (PCA) is performed to refine information and the meanings of the first principal components are interpreted, namely porosity and stretching direction as well as stretching factor. These properties can be estimated by finding the most similar known microstructures in the corresponding principal component space and are validated on both unseen algorithm-generated porous microstructures and reconstructed experimental microstructures. Second, Bayesian Optimization (BO) is utilized to efficiently solve the inverse problem of process–structure linkage, that is, finding the parameters for generating given target microstructures. The results are validated on several microstructures whose morphologies are different from the previous training data. Third, following the transfer learning strategy, a structure–property linkage (predicting the tortuosity of microstructure) is established by re-using the parts of the VAE model. The results show that its performance is better and more stable than the model trained from scratch, especially when only a small amount of data is provided. The above studies demonstrate the robust ability of the VAE model to characterize microstructures of porous membranes and have the potential for other geometries.

4.3 Enhanced analysis of characterization data

A detailed understanding of surface contamination and passivation compounds on Lithium Metal Anodes (LMAs) is crucial for their implementation in All-Solid-State Batteries (ASSBs). To characterize the surface condition of LMA samples, Time-of-Flight Secondary Ion Mass Spectrometry (ToF-SIMS), a highly sensitive surface analysis technique, can be utilized with reliability. Nonetheless, manual data analysis of complex ToF-SIMS data is often challenging and time-consuming. In this study, two well-established machine learning techniques are utilized to identify the characteristic secondary ions of five distinct pure lithium compounds, and their results are compared, namely, Principal Component Analysis (PCA) and Logistic Regression (LR). Moreover, both methods are applied to real LMAs samples and mixtures to enable the identification of their compositions based on the measured ToF-SIMS spectra. The machine learning-based approach demonstrates strong capabilities in identifying characteristic ions of measured compounds, which align with their chemical nature information. Furthermore, it achieves satisfactory accuracy in recognizing compositions of previously unseen samples. Additionally, the practical applications, scope, and limitations of this strategy are explored. This example demonstrates a sturdy analytical method that enables researchers to extract more information from their ToF-SIMS data more efficiently. This study has been published in [509] and the following sections are based on this publication. A conceptual illustration of this work is shown in Figure 4.14.

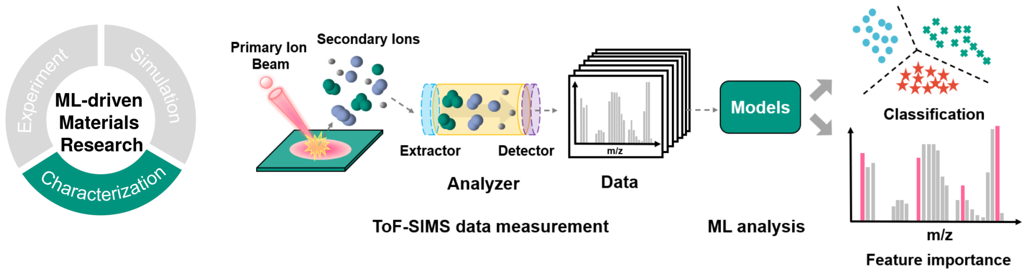


Figure 4.14: An overview of applying ML in analyzing characterization data of measured spectral data from lithium-compound materials using Time-of-Flight Secondary Ion Mass Spectrometry (ToF-SIMS). A classification model is developed to predict the classes of the studied compounds, based on which chemical information that matches the nature of the materials is extracted and analyzed from various perspectives. Reproduced with permission [509].

4.3.1 Materials and data

Sample preparation: 5 different lithium compounds, namely, Li_2CO_3, Li_2O, Li_3N, LiH and LiOH, were pressed into pellets (3 t, 1.5 min) under Ar-atmosphere. Mixtures of Li_2CO_3/LiOH and LiH/Li_2O were prepared by mixing equal weights of the selected lithium compounds in a vial by shaking and then following the same pressing. As reference LMA samples, one commercial lithium foil exposed to an N_2-plasma and two untreated foils were chosen. The LMA samples were cut into a small piece with a size of about 1×1 cm^2. All samples were attached to a LEICA sample holder above an electronically insulating tape. The subsequent transfer to the ToF-SIMS instrument was also done under Ar-atmosphere.

Data acquisition and management: the ToF-SIMS measurements were performed using a TOF.SIMS 5-100 instrument (ION-TOF GmbH, Muenster, Germany) equipped with a 25 kV Bi-cluster primary ion gun for analysis and with a 20 kV Gas Gluster Ion Beam (GCIB) for sputtering. For the pure lithium compounds, mixtures, and N_2-plasma sample, the corresponding surface regions were sputter-cleaned with ${Ar_{1500}}^+$ using a fluence of 4×10^{15} ions/cm^2 (10 kV, 10 nA) before the analysis. For the lithium foil, different fluences (1 frame = 9×10^{13} ions/cm^2 or 150 s = 9.5×10^{15} ions/cm^2) were applied to reach certain specific sample regions. For analysis, areas of 100×100 µm^2 were measured in spectrometry mode, using ${Bi_3}^+$ (20 kV, 1 pA) as primary ion by setting an ion dose density of 10^{12} ions/cm^2. Measurements were done in negative ion mode, with a cycle time of 100 µs. A flood gun was used for charge compensation during sputtering and analysis. For each material, 12 different spots were examined.

Data pre-processing: the software SurfaceLab 7.1 (ION-TOF GmbH, Muenster, Germany) was used for the first processing of the ToF-SIMS data. All spectra were calibrated to the ion signals $^6Li^-$, Li^-, O^-, OH^- and $O_2{}^-$. Afterward, an automated peak search was conducted up to mass-to-charge (m/z) = 120 a.u. and for peaks with a minimum of 100 counts, after which the borders of the found peaks were calibrated to uniform ranges for subsequent comparison. The peak area was exported for all selected peaks and spectra to be used for further analysis.

4.3.2 Analysis of data using principal component analysis

PCA workflow

Principal Component Analysis (PCA) is one of the most widely used techniques and represents the basis for many other methods [412]. Therefore, PCA is first adopted to analyze our data and its results are compared with those obtained by the logistic regression. General introductions to PCA [510, 249], as well as more specific studies related to PCA usage in ToF-SIMS can be found in literature [511, 512, 513]. The ToF-SIMS spectra are usually stacked to build the data matrix that is used as input for PCA, where each row of the matrix represents one sample spectrum and each column is a given m/z peak. PCA can be understood as a data transformation that combines a set of possibly correlated variables to a new set of uncorrelated variables, which is achieved by projecting the original data into a new orthogonal coordinate system where the axes are ordered according to the variance explained by each of these new axes, which are typically referred as Principal Components (PCs). The most important outputs of the PCA are scores and loadings. The scores are the projection of the original spectra on each PC, and each point in a scores plot stands for one measurement, that is, one ToF-SIMS spectrum. The loadings indicate the relationship between the principal components and the original features (peaks) of the data, that is, how the original features contribute to each PC. When applied in ToF-SIMS studies, for individual samples, the analysis of plotted scores and loadings can reveal which peaks allow for better discrimination of different samples. The NESAC/BIO toolbox (Spectragui) is used to perform PCA in this work.

Figure 4.15 gives an overview of how the measured 5 different lithium-containing compounds can be identified according to the ToF-SIMS peaks-based PCA scores and loadings. The analysis is based on the work of Graham and Castner and can be divided into three steps [412]. First, the different PCs are manually checked to select the ones that can best differentiate the 5 analyzed compounds, by giving priority to the analysis of the first PCs, which describe a higher variance of the data. For example, by checking the PC2, one can notice that LiH (red points in Figure 4.15a) has negative and much lower scores compared to the other compounds, making PC2 eligible as a discriminator between LiH and the other chemical species of interest. Next, the second step consists of analyzing the loadings of the identified PCs to identify which are the principal m/z peaks constituting that specific PC (in the example of Figure 4.15, PC2), i.e., which m/z peaks allow the distinction between the different compounds (in the same example, to distinguish between LiH and the other compounds). From the loading plot reported in Figure 4.15b it can be clearly seen that the peak contributing most to PC2 is found at $m/z = 9.03$, identifying it as the most important ion fragment for the identification of LiH. This inference is further verified by the peak area plot (Figure 4.15c) for the $m/z = 9.03$ peak, for which LiH shows the maximal intensity compared to the other compounds. Finally, the last step consists of identifying from which ion(s) originates the key peak(s) identified as a discriminator in steps 1 and 2, to assess the chemical correctness of this finding. In the example of Figure 4.15, the $m/z = 9.03$ peak is assigned to LiH_2^-, further justifying its usage to discriminate LiH from the other species. With the help of these steps, PCA can greatly reduce the complexity of ToF-SIMS analysis.

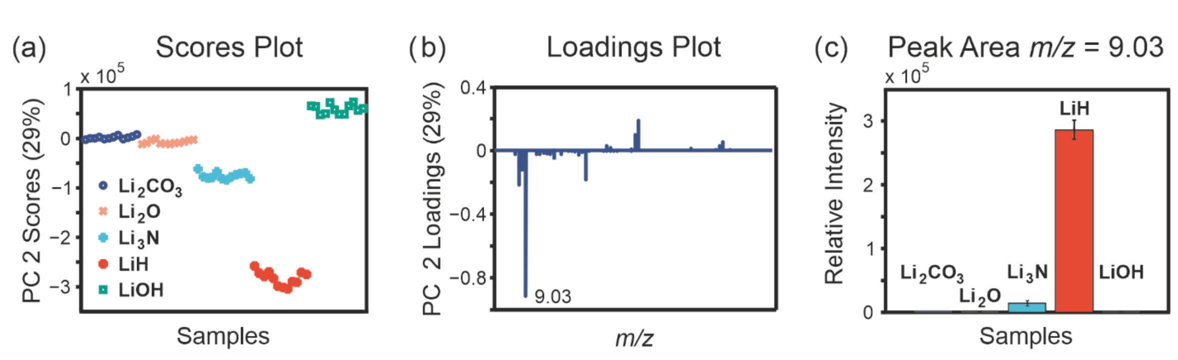


Figure 4.15: Example for the identification of key secondary ions for discriminating among the different Li compounds with PCA through a scores plot (a), and the corresponding loadings plot (b) and peak area plot (c). The scores and loadings plots are presented for PC2, explaining a data variance of 29%. Reproduced with permission [509].

Effects of different data pre-processing methods

In order to study the effects of different data pre-processing methods on PCA results, several processing methods are compared and the obtained scores plots are shown in Figure 4.16. The preferred pre-processing method is the one that benefits both the closer clustering of the same species and the clearer separation of different materials in the associated principal component space. It can be observed from the figure that if no pre-treatment is applied to the raw data, the PCA result is not satisfying: it is harder to distinguish some pure substances (e.g., Li_2CO_3 and Li_2O in scores plot PC1–2) as they overlap significantly. The situation is similar when using a logarithm scaling as Li_2CO_3 and LiOH are densely piled up in PC1–2, while Li_2O, Li_3N, and LiH overlap in PC3–4. For samples processed through normalization, the same substances are more closely grouped together, however, the distances between different groups are short, hindering a good classification. Results obtained from standardization, min-max scaling, and Poisson scaling methods are similar, with min-max scaling being slightly better than the other two as materials within the same group are clustered closer. In PC3–4, Li_2CO_3 and LiH can be better separated, and Li_3N and LiOH are clustered closely with standardization, while the opposite is true for min-max scaling. Poisson scaling does not have obvious advantages over others, though the 5 pure substances can be a bit better distinguished, in PC3–4, the mixture sample (Li_2CO_3+LiOH) will fully cover the area of LiH, which may lead to misjudgment. In summary, different pre-processing methods are found to have relatively complex influences on the PCA results and no one is found to perform much better than others in identifying all the species. However, among them, the min-max scaling shows comparatively better results, hence it is adopted across the study.

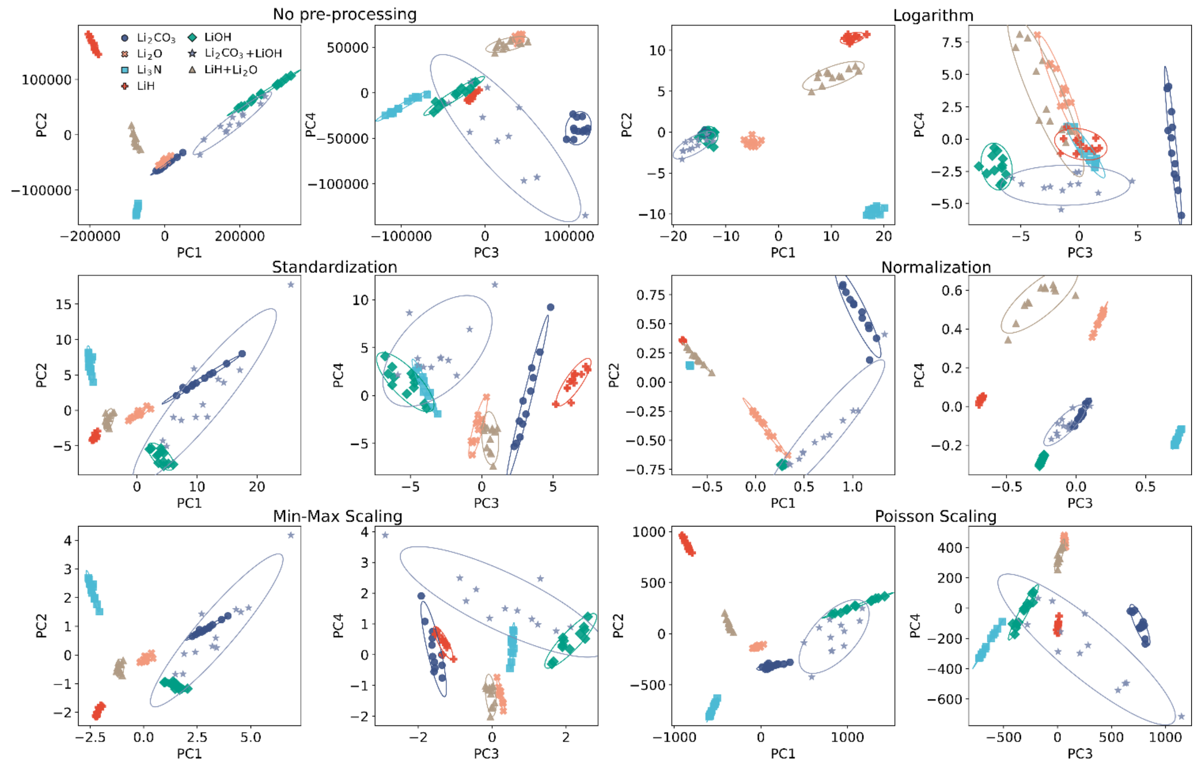


Figure 4.16: 2D scores plots of the first 4 PCs for the five pure lithium compounds and two mixture samples processed with different pre-processing methods: logarithmization (of one plus data to avoid undefined value), standardization (scaling data by removing the mean and scaling to unit variance), normalization (scaling each peak profile to its maximum), min-max scaling (scaling data in a range between 0 and 1 according to their minimum and maximum), Poisson scaling (accounting for Poisson noise in ToF-SIMS data). Reproduced with permission [509].

Identification of specific secondary ions

The characteristic (specific) peaks identified for the 5 lithium compounds through PCA are reported in Table 4.6A. The 15-top m/z values with the highest absolute loading values are chosen, i.e., the ones contributing the most to the identification of the different Li compounds. These peaks are then assigned to a specific ion fragment, and their area plots are checked. Table 4.6A reports the m/z and the assigned ion fragment of these peaks in order of their relevance (peak 1 with higher absolute loading than peak 2, and so forth). The results for both unscaled input data (Table 4.6A) and min-max scaled input data (Table 4.6B) are given.

As expected, it can be seen that the results based on scaled data are better, as there are more ion fragments marked in bold, indicating that they are chemically valid and reasonable specific secondary ions. Besides, checking the total intensities of the identified peaks shows that if the raw data are not pre-processed, only secondary ions with comparably high intensities can be found. In contrast, the min-max scaling can mitigate this effect by allowing peaks with lower intensities to be treated equally and hence can facilitate the analysis. In summary, a proper pre-processing of the data is a necessity for performing robust analysis.

For a better comparison between PCA and the Logistic Regression (LR) model (detailed in a later section), the m/z values identified by both techniques are underlined in Table 4.6. It is found that results obtained for Li_2CO_3, Li_3N, LiH, and LiOH by the two methods are very similar: both techniques have identified at least 8 out of 15 common characteristic peaks. A few peaks also match chemical information well despite their relatively low intensity (marked in normal font). Moreover, almost no secondary ions are found to mismatch their chemical information (marked in italics). Overall, the validity of the two methods is further demonstrated by the similarities between their results and the good compatibility with the chemical nature of each tested compound. However, access to the key information is more easily accessible with the LR model than with PCA, which requests manual steps to identify these characteristic peaks.

Identification of compositions for mixture samples

PCA is tested to identify the 5 lithium compounds on the mixture and LMA samples. The presence (or absence) of each lithium compound in the samples of interest is inferred according to the score plots of the first several PCs, i.e., the ones describing most of the data information. Specifically, the PCs of the samples of interest and pure reference compounds are compared: if the PCs of the interested sample are inside the confidence interval (here 95% confidence interval) of the PCs value associated with one reference Li compound, it is inferred that it contained that specific compound. To illustrate and verify such a procedure, two kinds of powder mixtures, namely 1:1 weight mixed $LiOH+Li_2CO_3$ (sample No. 1) and 1:1 $LiH+Li_2O$ (sample No. 2), are prepared and analyzed through PCA. For a fair comparison, PCA is first performed on pure lithium compounds to obtain the base principal components and later mapped to the two mixture samples. The

Table 4.6: Results of identified specific peaks presented in different font styles for better illustration. Besides, m/z values identified by both PCA and LR techniques (with min-max scaled data) are underlined for better comparison.

A) PCA no scaling

	LiH		Li_3N		Li_2CO_3		LiOH		Li2O	
No.	m/z	Ion	m/z	Ion	m/z	Ion	m/z	Ion	m/z	Ion
1	9.03	**LiH2-**	26.00	**CN-**	67.00	$\mathbf{LiCO_3}^-$	41.02	$\mathbf{LiO_2H_2}^-$	7.02	Li^-
2	7.02	Li^-	7.02	Li^-	59.99	$\mathbf{CO_3}^-$	40.01	$\mathbf{LiO_2H}^-$	25.01	C_2H^-
3	26.00	CN^-	16.02	**NH2-**	31.99	O_2^-	7.02	Li^-	13.01	CH^-
4	8.02	$\mathbf{LiH}^-$	15.01	$\mathbf{NH}^-$	39.00	$\mathbf{LiO_2}^-$	65.04	$\mathbf{Li_2O_3H_3}^-$	23.01	LiO^-
5	8.03	$\mathbf{^6LiH_2}^-$	12.00	C^-	40.01	LiO_2H^-	31.99	$\mathbf{O_2}^-$	6.02	$^6Li^-$
6	17.05	$\mathbf{Li_2H_3}^-$	6.02	$^6Li^-$	23.01	$\mathbf{LiO}^-$	9.03	$\mathit{LiH_2}^-$	19.00	F^-
7	24.02	$LiOH^-$	67.00	$\mathit{LiCO_3}^-$	12.00	$\mathbf{C}^-$	64.04	$\mathbf{Li_2O_3H_2}^-$	14.03	Li_2^-
8	24.00	C_2^-	8.02	LiH^-	7.02	Li^-	26.00	CN^-	19.01	H_3O^-
9	13.01	CH^-	14.03	$\mathbf{Li_2}^-$	13.01	CH^-	33.00	$\mathbf{O_2H}^-$	31.02	CH_3O^-
10	16.02	$\mathit{NH_2}^-$	59.99	$\mathit{CO_3}^-$	25.01	C_2H^-	39.00	LiO_2^-	41.00	C_2HO^-
11	19.00	F^-	13.01	CH^-	31.02	$\mathbf{CH_3O}^-$	67.00	$\mathit{LiCO_3}^-$	24.00	C_2^-
12	12.00	C^-	25.01	C_2H^-	45.00	$\mathbf{CO_2H}^-$	56.01	$\mathbf{LiO_3H}^-$	23.02	$^6LiOH^-$
13	6.02	$^6Li^-$	24.00	C_2^-	55.00	$\mathbf{LiO_3}^-$	25.03	$\mathbf{LiOH_2}^-$	18.00	H_2O^-
14	15.04	$\mathbf{Li_2H}^-$	23.01	LiO^-	66.00	$\mathbf{^6LiCO_3}^-$	23.01	LiO^-	22.01	$^6LiO^-$
15	14.03	Li_2^-	22.02	$\mathbf{LiNH}^-$	25.03	$LiOH_2^-$	24.02	$LiOH^-$	49.01	C_4H^-

B) PCA min-max scaling

	LiH		Li_3N		Li_2CO_3		LiOH		Li_2O	
No.	m/z	Ion	m/z	Ion	m/z	Ion	m/z	Ion	m/z	Ion
1	$\underline{14.04}$	$\mathbf{6\,LiLiH}^-$	$\underline{21.02}$	$\mathbf{LiN}^-$	$\underline{67.00}$	$\mathbf{LiCO_3}^-$	$\underline{34.00}$	$\mathbf{H_2O_2}^-$	19.00	F^-
2	$\underline{9.03}$	$\mathbf{LiH_2}^-$	$\underline{22.02}$	$\mathbf{LiNH}^-$	$\underline{66.00}$	$\mathbf{^6LiCO_3}^-$	$\underline{64.04}$	$\mathbf{Li_2O_3H_2}^-$	31.02	CH_3O^-
3	$\underline{8.03}$	$\mathbf{6\,LiH_2}^-$	$\underline{14.03}$	$\mathbf{Li_2}^-$	$\underline{43.99}$	$\mathbf{CO_2}^-$	$\underline{33.00}$	$\mathbf{O_2H}^-$	31.03	Li_2OH^-
4	$\underline{15.04}$	$\mathbf{Li_2H}^-$	$\underline{13.03}$	$\mathbf{^6LiLi}^-$	$\underline{59.99}$	$\mathbf{CO_3}^-$	$\underline{19.01}$	$\mathbf{H_3O}^-$	55.04	$CH_4O_2Li^-$
5	$\underline{22.06}$	$\mathbf{Li_3H}^-$	$\underline{26.00}$	$\mathbf{CN}^-$	$\underline{46.03}$	$\mathbf{Li_2O_2}^-$	$\underline{18.01}$	$\mathbf{H_2O}^-$	43.02	C_3Li^-
6	$\underline{17.05}$	$\mathbf{Li_2H_3}^-$	$\underline{15.01}$	$\mathbf{NH}^-$	31.02	CH_3O^-	$\underline{65.04}$	$\mathbf{Li_2O_3H_3}^-$	19.01	H_3O^-
7	$\underline{25.08}$	$\mathbf{Li_3H_4}^-$	$\underline{27.00}$	$\mathbf{CHN}^-$	$\underline{55.00}$	$\mathbf{LiO_3}^-$	$\underline{56.01}$	$\mathbf{LiO_3H}^-$	7.02	Li^-
8	$\underline{24.09}$	$\mathbf{^6LiLi_2H_4}^-$	$\underline{21.05}$	$\mathbf{Li_3}^-$	38.01	$^6LiO_2^-$	$\underline{41.02}$	$\mathbf{LiO_2H_2}^-$	6.02	$^6Li^-$
9	$\underline{33.10}$	$\mathbf{Li_4H_5}^-$	6.02	$^6Li^-$	23.02	$^6LiOH^-$	$\underline{63.03}$	$\mathbf{Li_2O_3H}^-$	14.03	Li_2^-
10	$\underline{16.05}$	$\mathbf{Li_2H_2}^-$	7.02	Li^-	$\underline{23.01}$	$\mathbf{LiO}^-$	$\underline{40.01}$	$\mathbf{LiO_2H}^-$	45.03	$C_2H_5O^-$
11	$\underline{16.05}$	$\mathbf{6\,LiLiH_3}^-$	$\underline{23.04}$	$\mathbf{LiNH_2}^-$	55.04	$CH_4O_2Li^-$	6.02	$^6Li^-$	13.03	$^6LiLi^-$
12	$\underline{7.02}$	$\mathbf{^6LiH}^-$	$\underline{30.00}$	$\mathbf{NO}^-$	15.02	CH_3^-	7.02	Li^-	31.01	NOH^-
13	21.05	Li_3^-	31.01	NOH^-	22.01	$6\,LiO^-$	$\underline{49.05}$	$\mathbf{Li_2O_2H_3}^-$	15.02	CH_3^-
14	$\underline{8.02}$	$\mathbf{LiH}^-$	$\underline{16.02}$	$\mathbf{NH_2}^-$	$\underline{39.00}$	$\mathbf{LiO_2}^-$	$\underline{63.96}$	O_4^-	18.01	H_2O^-
15	14.03	Li_2^-	$\underline{24.04}$	$\mathbf{LiNH_3}^-$	$\underline{60.99}$	$\mathbf{CO_3H}^-$	47.97	O_3^-	23.02	$^6LiOH^-$

C) LR min-max scaling

	LiH		Li_3N		Li_2CO_3		LiOH		Li_2O	
No.	m/z	Ion	m/z	Ion	m/z	Ion	m/z	Ion	m/z	Ion
1	$\underline{9.03}$	$\mathbf{LiH_2}^-$	$\underline{27.00}$	$\mathbf{CHN}^-$	$\underline{67.00}$	$\mathbf{LiCO_3}^-$	$\underline{65.04}$	$\mathbf{Li_2O_3H_3}^-$	48.00	C_4^-
2	$\underline{8.03}$	$\mathbf{6\,LiH_2}^-$	$\underline{26.00}$	$\mathbf{CN}^-$	$\underline{66.00}$	$\mathbf{^6LiCO_3}^-$	$\underline{64.04}$	$\mathbf{Li_2O_3H_2}^-$	49.01	C_4H^-
3	$\underline{17.05}$	$\mathbf{Li_2H_3}^-$	$\underline{21.02}$	$\mathbf{LiN}^-$	$\underline{43.99}$	$\mathbf{CO_2}^-$	$\underline{49.05}$	$\mathbf{Li_2O_2H_3}^-$	49.05	$Li_2O_2H_3^-$
4	$\underline{25.08}$	$\mathbf{Li_3H_4}^-$	$\underline{22.02}$	$\mathbf{LiNH}^-$	$\underline{59.99}$	$\mathbf{CO_3}^-$	$\underline{63.03}$	$\mathbf{Li_2O_3H}^-$	8.02	LiH^-
5	$\underline{16.05}$	$\mathbf{Li_2H_2}^-$	$\underline{15.01}$	$\mathbf{NH}^-$	12.00	$\mathbf{C}^-$	$\underline{34.00}$	$\mathbf{H_2O_2}^-$	31.99	O_2^-
6	$\underline{24.09}$	$\mathbf{^6LiLi_2H_4}^-$	$\underline{23.04}$	$\mathbf{LiNH_2}^-$	68.01	$\mathbf{LiCO_3H}^-$	$\underline{41.02}$	$\mathbf{LiO_2H_2}^-$	65.04	$Li_2O_3H_3^-$
7	$\underline{16.05}$	$\mathbf{^6LiLiH_3}^-$	$\underline{30.00}$	NO^-	54.00	$\mathbf{^6LiO_3}^-$	$\underline{33.00}$	$\mathbf{O_2H}^-$	15.04	Li_2H^-
8	$\underline{15.04}$	$\mathbf{Li_2H}^-$	34.03	$\mathbf{CHNLi}^-$	51.01	$H_3O_3^-$	$\underline{56.01}$	$\mathbf{LiO_3H}^-$	56.01	LiO_3H^-
9	$\underline{33.10}$	$\mathbf{Li_4H_5}^-$	$\underline{16.02}$	$\mathbf{NH_2}^-$	$\underline{60.99}$	$\mathbf{CO_3H}^-$	$\underline{40.01}$	$\mathbf{LiO_2H}^-$	60.99	CO_3H^-
10	$\underline{8.02}$	$\mathbf{LiH}^-$	33.02	$CNLi^-$	$\underline{55.00}$	$\mathbf{LiO_3}^-$	$\underline{18.01}$	H_2O^-	36.00	C_3^-
11	$\underline{7.02}$	$\mathbf{^6LiH}^-$	$\underline{24.04}$	$\mathbf{LiNH_3}^-$	61.99	$CO_3H_2^-$	$\underline{19.01}$	H_3O^-	64.04	$Li_2O_3H_2^-$
12	$\underline{22.06}$	$\mathbf{Li_3H}^-$	$\underline{21.05}$	$\mathbf{Li_3}^-$	$\underline{39.00}$	$\mathbf{LiO_2}^-$	33.99	$\mathbf{H_2O_2}^-$	12.00	C^-
13	33.05	$Li_2OH_3^-$	38.04	CH_3OLi^-	62.02	$\mathbf{Li_2O_3}^-$	31.99	$\mathbf{O_2}^-$	63.03	$Li_2O_3H^-$
14	$\underline{14.04}$	$\mathbf{^6LiLiH}^-$	$\underline{14.03}$	$\mathbf{Li_2}^-$	38.01	$\mathbf{^6LiO_2}^-$	47.03	$\mathbf{Li_2O_2H}^-$	33.99	$\mathit{PH_3}^-$
15	24.00	C_2^-	$\underline{13.03}$	$\mathbf{^6LiLi}^-$	$\underline{46.03}$	$\mathbf{Li_2O_2}^-$	$\underline{63.96}$	O_4^-	55.00	LiO_3^-

resulting scores plots of the first four PCs are shown in Figure 4.17. The results illustrate that the aforementioned procedure works well with sample No. 1: LiOH+Li_2CO_3 mixture, as its scores partially overlap with regions identified by the pure LiOH and Li_2CO_3 in PC1–2 (Figure 4.17a), while the scores in PC3–4 (Figure 4.17b) are distributed along the line of concatenation between the pure LiOH and Li_2CO_3. Therefore, it can be inferred that mixture sample No. 1 contains both LiOH and Li_2CO_3. However, for mixture sample No. 2 (composed of LiH and Li_2O), the interpretation is less straightforward. In the scores plot of PC1 vs. PC2, this sample does not overlap with any regions of LiH or Li_2O. In the scores plot of PC3 vs. PC4, data points of mixture sample No. 2 overlap with the region of Li_2O, but are far away from all other compounds, and notably far from LiH. Based on these results, it would not have been possible to correctly identify LiH in this mixture sample using PCA. This indicates that PCA cannot always be used to correctly identify the compositions of the mixture, especially when multiple components could interfere with the identification. Besides, even if such kind of visualization method is very intuitive, it requires the researcher to manually check the results and choose which are the reference samples in the mixture. This procedure could be particularly challenging if the analysis of different PCs leads to different conclusions. Given all the above, PCA does not seem to be suitable for the identification of the composition of a mixture.

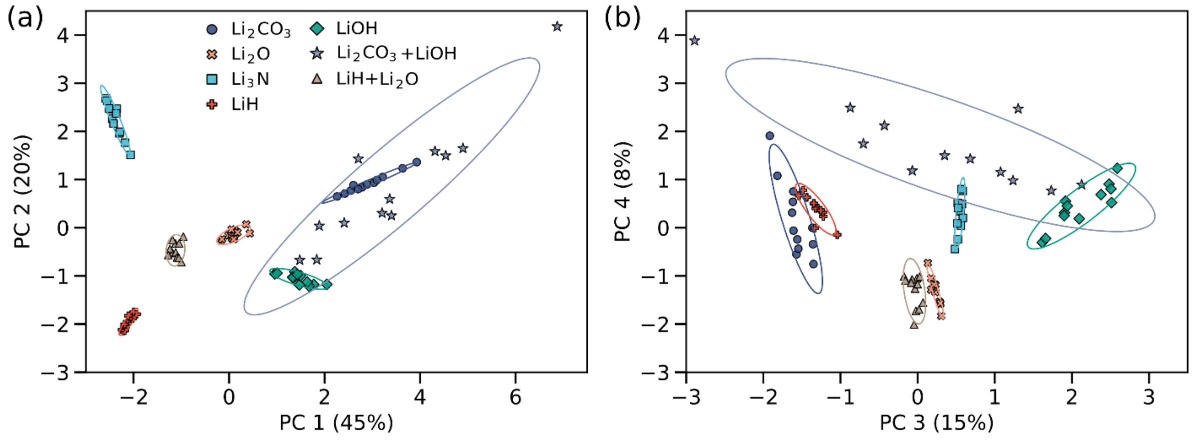


Figure 4.17: 2D scores plot of the first four PCs for the 5 pure lithium compounds and 2 mixtures (Li_2CO_3+LiOH noted as sample No. 1 and LiH+Li_2O noted as sample No. 2). The PCs are numbered according to the proportions of variances (listed in parentheses) they include starting with PC 1 for the highest value. Reproduced with permission [509].

Results of pure lithium compounds and LMA samples

PCA results of five measured pure lithium compounds and LMA samples No. 3–5 in the main text are shown in Figure 4.18. From these scores plots, it can be seen that data points of sample No. 5 lie between several pure substances in PC1–2 (Figure 4.18a) and overlap with Li_2CO_3 and LiH in PC3–4 (Figure 4.18b). Such results can easily mislead the identification of its compositions and a lot of manual interventions are required to classify these samples. The case is more complex for sample No. 3 as it is far away from those of pure substances in principal component spaces, hindering the determination of its compositions. In particular, for sample No. 4, its far apart from other samples suggests its unusual data distribution, implying a further examination and the need to deal with this outlier. From these results, it can be seen that using PCA alone to infer the composition of the mixture accurately is difficult. Therefore, alternative ML methods are tested and the LR model is finally selected as the candidate to overcome the limits of PCA, as it does not only perform classification but also outputs the probability of belonging to a given class, which can be used to infer the possible components in mixtures with minimal manual interpretation.

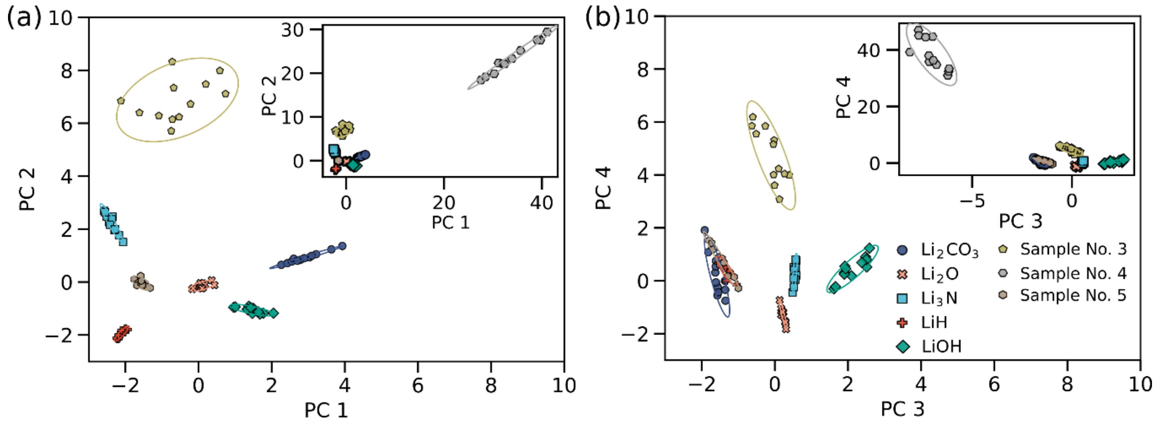


Figure 4.18: 2D scores plots of the first four PCs for the five pure lithium compounds and three LMA samples No. 3–5. For better visualization of all data points, sample No. 4 is shown separately in the embedded plot of each sub-figure in (a) and (b). Reproduced with permission [509].

4.3.3 Analysis of data using logistic regression

The schematic diagram of the workflow followed to build the Logistic Regression (LR) model is illustrated in Figure 4.19, and it mainly consists of four processing steps (from left to right in the figure):

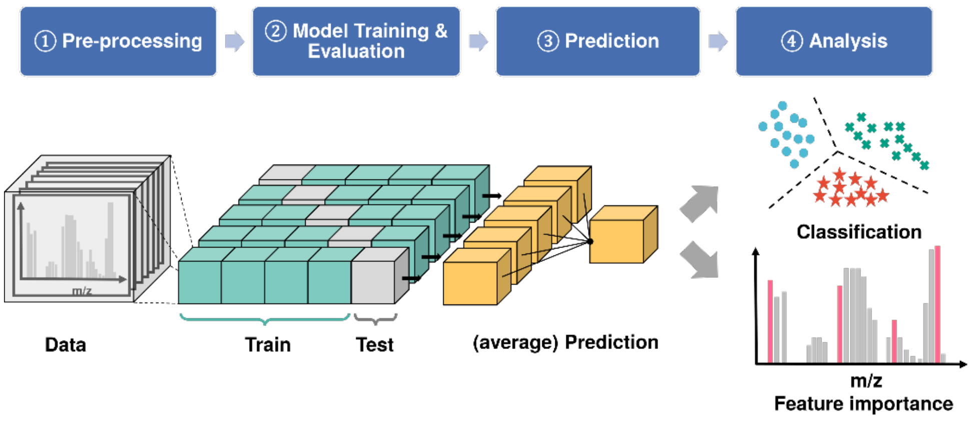


Figure 4.19: Schematic workflow adopted to build the LR model applied to the analysis of ToF-SIMS data, which consists of data pre-processing, model training, 5-fold cross-validation to evaluate its performance, application on unseen samples (predictions), and analysis of the classification results. More information about ToF-SIMS spectra, such as the importance of specific peaks or the composition of the mixture samples, can be obtained by further analysis of the model results. Reproduced with permission [509].

1. Pre-processing: the raw ToF-SIMS data are denoised and scaled to have comparable values between different variables, which is one of the key steps preceding any ML-based approach. In the field of ToF-SIMS analysis, there are many commonly used pre-processing methods. However, no general method was found yet to be able to fit all the situations. In this study, the min-max scaling method is found to perform well on this data set. For the min-max scaling, each intensity value of spectra is first subtracted from the minimum value of the corresponding m/z peak and is then divided by its intensity range (difference between the original maximum and minimum intensity of that m/z peak). In this way, the intensity in each m/z peak is scaled to a range between 0 and 1. This kind of scaling approach was found to be beneficial for the identification of the specific secondary ions, allowing the distinction between the samples.

2. Model training and evaluation: the LR model is trained to classify the 5 pure lithium compounds based on their measured ToF-SIMS spectra. An accurate model should not only be able to classify samples it has been trained on, but it should also perform well on new samples that it has not seen before. The K-fold Cross-Validation (CV) strategy is used to take full advantage of the limited amount of ToF-SIMS data and to assess the predictive performance of the LR model in a fair manner. The prediction accuracy is quantified as the number of correct predictions divided by the total one, and through the F-score. The K evaluation results are then averaged to assess the overall performance of the model. Besides, in many cases, an optional extra unseen test data set can also be utilized to further evaluate

the model performance. In this study, there is no such kind of additional test set due to the limited number of measured samples. A 5-fold cross-validation scheme is adopted, and in each cross-validation step, the min-max scaling is performed. The scaling parameters are defined as a function of the training set, and these same parameters are applied to the associated test set. This allows keeping the test set as "unseen" for the model, which ensures the fairness of the model evaluation procedure. Considering the limited amount of training data (60 spectra from 5 pure compounds) and the large variance within the data, this process is critical to ensure robust results.

3. Prediction: as mentioned above, the fitted LR model is evaluated on 5 test sets (12 spectra each) of pure Li compounds through a 5-fold CV, to ensure its correctness. Once the LR model accuracy for pure Li compounds is assessed, it is further applied to compound mixtures, to test its capability of identifying their compositions, and to real LMA samples. In this case, the probability of each sample belonging to any pure Li compound is computed through the trained LR model, and it is considered that a given Li compound is present when the associated probability is high enough to suggest this.

4. Analysis: to better analyze and interpret the results of the LR model, the data of ToF-SIMS spectra were reduced to low dimensions with the aid of PCA to visualize the classification results from the LR model. Moreover, feature importance obtained from the LR model's parameters is utilized to infer which are the specific ions allowing to better distinguish among different compounds. The final identification of the key specific ions is based on the average feature importance calculated during the 5-fold CV.

Comparison of different classification models

Table 4.7 summarizes the results of classifying the 5 measured pure lithium compounds from commonly utilized machine learning algorithm architectures. To evaluate the performance of different models, two measures are calculated, namely, accuracy and F-score metrics, and the higher their scores, the better the model performs [514]. Trained on pure compounds only, these models are further applied to two mixture samples (LiH/Li_2O and $Li_2CO_3/LiOH$, mixing in equal weight proportions, respectively), and their predicted probability values are also shown in Table 4.7 where the two largest values are bolded. The probability value indicates how confident the model is in the prediction of individual compounds. For simplification, the model is expected to output a certain probability value for the corresponding components in the mixture and zero for others. It should be noted that the probability value can indicate how confident the model is in the prediction of the individual compound and these probability values are used only as a qualitative indicator, which means, they do not coincide strictly with their proportions in the mixture. The predictions may deviate from the expected values due to many possible reasons, such as uneven mixing and the matrix effects, which will change the assumption of a linear combination of two components and therefore can disturb the model.

In this study, the tested machine learning models are: logistical regression [49], Support Vector machine Classifier (SVC) with linear or Radial Basis Function (RBF) kernel [52], k-Nearest Neighbors (k-NN) (reported in an unpublished US Air Force School of Aviation Medicine report in 1951 and later introduced as a non-parametric algorithm for pattern classification) [515], decision tree [58], and its advanced variants: random forest [72, 65] and Adaptive Boosting (AdaBoost) method (both with decision tree as the base estimator) [67], Gaussian Naive Bayes (NB) [516], and a shallow neural network with Multi-Layer Perceptron (MLP) [108]. Each model performs well when trained on pure substances and applied in identifying pure Li compounds. On the one hand, it is satisfying that these models can achieve good performance with 98–100% accuracy/F-score. On the other hand, such a high evaluation score poses difficulties in tuning the hyperparameters for these models. As a result, for simplicity, the default parameter settings from the associated machine learning library are adopted for all the models. As can be seen from Table 4.7, when the models are applied to mixture samples, some models show strong overfitting and fail to identify the correct multi-compositions (e.g., k-NN model fails to detect LiH in LiH/Li_2O mixture) or cannot distinguish different substances well (e.g., SVC with RBF kernel just yields relatively uniform probability values for all the substances). This phenomenon is understandable because the data distributions of mixture samples can be quite different from those of pure substances. In this case, the LR model performs better than the others. It is speculated that such a good performance comes from the LR model's nature of being a quasi-linear model: as the spectra of mixtures can be somehow seen as a (possible) linear combination of the spectra of pure substances and hence the LR model suits better on these samples.

It is important to note that the comparison between models is neither systematic nor comprehensive. Tuning the hyperparameters of these models is likely to produce very different results. As an example, for the LR model, different hyperparameters are tried to see their effect on the model performance, and their results are listed in Table 4.8. To be specific, here the hyperparameters are mainly related to the selection of regularization terms and different regularization strengths, that is, no regularization, L_1 or L_2 regularization, or elastic net (both L_1 and L_2, here their ratio is set to 1:1 for simplicity). It can be seen that the model without regularization term has already received a satisfying result, though it yields a noteworthy prediction of possible (10% probability) existence of Li_3N (which ought not to be there) for the LiH/Li_2O mixture sample. On the one hand, it can also be noticed that a too strong regularization (smaller value of C) of the model is detrimental to its performance. On the other hand, an appropriate regularization strength can ensure the good performance of the model, and no significant differences are observed in terms of classifying pure compounds (about 100% accuracy/F-score) with moderate regularization settings. Since it is also desired to use the model to determine the composition of the mixture, the model should ideally also predict higher probability values for the substances that it may contain. With this rule, combinations of hyperparameters that give unreasonable prediction values can be excluded. For example, with L_1 regularization ($C = 1$), although the classification performance is almost perfect for pure substances, it gives high predicted probability values for substances that should not be present in the mixture samples (e.g., 0.18 for Li_2CO_3, 0.19 for Li_3N and 0.19 for LiOH in LiH+Li_2O mixture sample, and 0.19 for Li_2O, 0.16 for Li_3N and 0.15 for LiH in Li_2CO_3+LiOH mixture sample). Since the predicted probability values do not strictly quantify the proportion of components in the mixture (see main text for detailed discussion), it is hard to choose the optimal one among the remaining moderate hyperparameters. In this regard, the model trained with the default hyperparameters (L_2 regularization with strength $C = 1$) from the machine learning library is empirically chosen. However, such kind of study of all the combined parameters for all these models would be too time-consuming and tedious. Hence, the principle of Occam's razor is followed: choosing a simple and practical model. Therefore, the LR model is finally selected as the standard model in this study.

Identification of specific secondary ions

The trained LR model is able to make accurate (>99%) classifications of the measured 5 kinds of pure Li compound samples. Based on the feature importance given by the LR model, the characteristic peaks are identified for the 5 lithium compounds samples and they are listed in Table 4.6C, where the 15-top m/z values are reported in the order of importance. For Li_2CO_3, Li_3N, LiH, and LiOH, the LR model has identified at least 12 out of 15 characteristic peaks (marked in bold font) that have the highest intensities in the peak area plots and shows good compatibility with their corresponding chemical nature. A few peaks also match chemical information despite their relatively low intensity (marked in normal font). Moreover, almost no secondary ions are found to mismatch their chemical information (marked in italics).

Besides, it is worth noting that the result for Li_2O is unsatisfying, as the selective identification of Li_2O through ToF-SIMS can be particularly difficult due to its similarities with Li_2CO_3. For example, some peaks, such as LiO^- and $LiO_2{}^-$, which could be intuitively regarded as specific for Li_2O, appear even more intense in the Li_2CO_3 samples. As a result, no prominent secondary ions solely specific to Li_2O can be determined by the LR

Table 4.7: Accuracy and F-score for classifying pure Li compounds and predicted probability for mixture samples from different machine learning models.

Model	Pure compounds		Mixture	Composition probability				
	Accuracy	F-score		Li_2CO_3	Li_2O	Li_3N	LiH	LiOH
LR	1.0	1.0	LiH + Li_2O	0.00	0.83	0.03	0.14	0.00
			Li_2CO_3+ LiOH	0.54	0.00	0.00	0.00	0.46
SVC (RBF)	1.0	1.0	LiH + Li_2O	0.09	0.48	0.15	0.20	0.08
			Li_2CO_3+ LiOH	0.33	0.15	0.11	0.11	0.30
SVC (linear)	1.0	1.0	LiH + Li_2O	0.05	0.51	0.16	0.21	0.07
			Li_2CO_3+ LiOH	0.44	0.11	0.03	0.02	0.39
k-NN	1.0	1.0	LiH + Li_2O	0.00	0.99	0.00	0.01	0.00
			Li_2CO_3+ LiOH	0.50	0.00	0.00	0.00	0.50
Decision Tree	0.97	0.97	LiH + Li_2O	0.00	0.10	0.57	0.33	0.00
			Li_2CO_3+ LiOH	0.62	0.00	0.00	0.00	0.38
Random Forest	1.0	1.0	LiH + Li_2O	0.03	0.21	0.28	0.42	0.06
			Li_2CO_3+ LiOH	0.37	0.08	0.02	0.00	0.53
AdaBoost	0.97	0.97	LiH + Li_2O	0.00	0.98	0.00	0.02	0.00
			Li_2CO_3+ LiOH	0.37	0.08	0.02	0.00	0.53
Gaussian NB	1.0	1.0	LiH + Li_2O	0.00	0.00	0.00	1.00	0.00
			Li_2CO_3+ LiOH	0.08	0.00	0.00	0.00	0.92
MLP	1.0	1.0	LiH + Li_2O	0.06	0.39	0.16	0.32	0.07
			Li_2CO_3+ LiOH	0.41	0.03	0.00	0.00	0.56

Table 4.8: Accuracy and F-score for classifying pure Li compounds and predicted probability for mixture samples from LR models with different hyperparameter settings. (C: Inverse of regularization strength)*

Regularization	C^*	Pure compounds		Mixture	Composition probability				
		Accuracy	F-score		Li_2CO_3	Li_2O	Li_3N	LiH	LiOH
None	-	1.0	1.0	LiH + Li_2O	0.00	0.74	0.04	0.21	0.01
				Li_2CO_3 + LiOH	0.50	0.01	0.00	0.00	0.49
L_1	0.01	0.13	0.04	LiH + Li_2O	0.20	0.20	0.20	0.20	0.20
				Li_2CO_3 + LiOH	0.20	0.20	0.20	0.20	0.20
	0.1	0.12	0.04	LiH + Li_2O	0.20	0.20	0.20	0.20	0.20
				Li_2CO_3 + LiOH	0.20	0.20	0.20	0.20	0.20
	1	1.0	1.0	LiH + Li_2O	0.06	0.37	0.08	0.40	0.08
				Li_2CO_3 + LiOH	0.42	0.09	0.05	0.02	0.42
	10	1.0	1.0	LiH + Li_2O	0.01	0.53	0.05	0.38	0.02
				Li_2CO_3 + LiOH	0.50	0.05	0.00	0.00	0.45
	100	1.0	1.0	LiH + Li_2O	0.00	0.59	0.04	0.36	0.00
				Li_2CO_3 + LiOH	0.51	0.03	0.00	0.00	0.46
L_2	0.01	0.92	0.92	LiH + Li_2O	0.15	0.24	0.22	0.24	0.16
				Li_2CO_3 + LiOH	0.30	0.16	0.12	0.11	0.30
	0.1	1.0	1.0	LiH + Li_2O	0.01	0.69	0.09	0.20	0.02
				Li_2CO_3 + LiOH	0.53	0.06	0.01	0.00	0.45
	1	1.0	1.0	LiH + Li_2O	0.00	0.83	0.03	0.14	0.00
				Li_2CO_3 + LiOH	0.54	0.00	0.00	0.00	0.46
	10	1.0	1.0	LiH + Li_2O	0.00	0.89	0.03	0.08	0.00
				Li_2CO_3 + LiOH	0.55	0.00	0.00	0.00	0.45
	100	1.0	1.0	LiH + Li_2O	0.00	0.93	0.02	0.05	0.00
				Li_2CO_3 + LiOH	0.46	0.00	0.00	0.00	0.54
Elastic net	0.01	0.92	0.92	LiH + Li_2O	0.19	0.19	0.20	0.20	0.21
				Li_2CO_3 + LiOH	0.19	0.19	0.20	0.20	0.21
	0.1	1.0	1.0	LiH + Li_2O	0.18	0.23	0.19	0.21	0.19
				Li_2CO_3 + LiOH	0.26	0.19	0.16	0.15	0.24
	1	1.0	1.0	LiH + Li_2O	0.07	0.54	0.10	0.21	0.07
				Li_2CO_3 + LiOH	0.49	0.10	0.01	0.01	0.38
	10	1.0	1.0	LiH + Li_2O	0.05	0.50	0.12	0.28	0.06
				Li_2CO_3 + LiOH	0.47	0.07	0.00	0.00	0.45
	100	1.0	1.0	LiH + Li_2O	0.04	0.48	0.15	0.29	0.04
				Li_2CO_3 + LiOH	0.52	0.06	0.00	0.00	0.42

model within this sample data set.

Identification of compositions for mixtures and LMAs

The identified specific peaks for the different lithium compounds can help researchers recognize these materials from their ToF-SIMS spectra, especially for those having unique characteristic peak(s). In addition, it is desirable to be able to identify chemical species with no particular characteristic peak, like Li_2O in the present samples. However, realistic samples may contain several different compounds, from which secondary ions could originate that overlap on the characteristic peaks identified for the pure Li compounds under consideration. This could significantly complicate the spectrometric analysis. To identify the compositions on a mixture sample, the obtained LR model is used to infer the possible components based on the probability values it outputs for different substances.

Identification of compositions for mixtures with LR

The obtained LR model does not only perform classification but also outputs the probability of belonging to a given class, which can be used to infer the possible components in mixtures without much need for manual interpretation. To better understand the predicted probability values from the LR model and the work process of the model, 5 pure lithium compounds and 2 mixture samples (Li_2CO_3+LiOH and LiH+Li_2O) are projected to the low-dimension space with PCA for better visualization (shown in Figure 4.20a and 4.20c). It can be seen that both mixtures lie in the position between their corresponding two constituents. Besides, the decision boundary plots of LiH are also shown in Figures 4.20b and 4.20d. They serve as the schematic diagram to illustrate how the compositions of a Li_2O+LiH mixture sample can be identified by the LR model, i.e., the probability associated with its presence. These decision boundary plots are obtained as follows: the PC1–3 spaces are uniformly gridded into dense data points, which are then inversely mapped back to the original feature space (m/z peaks), and their corresponding probabilities of being LiH are calculated by the LR model. The inverse mapping is performed in this way: PCA scores $\boldsymbol{S}$ can be given by $\boldsymbol{S} = \boldsymbol{X}\boldsymbol{W}$, where $\boldsymbol{X}$ is the raw (centered) data matrix and $\boldsymbol{W}$ is the matrix consisting of eigenvectors, therefore, the inverse reconstruction of data $\hat{\boldsymbol{X}}$ to the original feature space is calculated by $\hat{\boldsymbol{X}} \approx \boldsymbol{S}\boldsymbol{W}^T$. In the schematic diagram, only pure LiH (red star), Li_2O (blue cross) substances, and the associated mixture samples (purple triangle) are shown for the convenience of visualization. The warmer the color, the higher the probability that LiH is present according to the model, while cold color indicates a low probability. It can be clearly seen that the mixtures lie in between the two reference compounds and are biased towards Li_2O (especially in PC2–3), which is consistent with the predicted probability values (0.83 for Li_2O and 0.14 for LiH) for this mixture sample (sample No. 2 in Table 4.9). In conclusion, for mixture samples, their corresponding compositions can be estimated qualitatively based on the LR model's predictions.

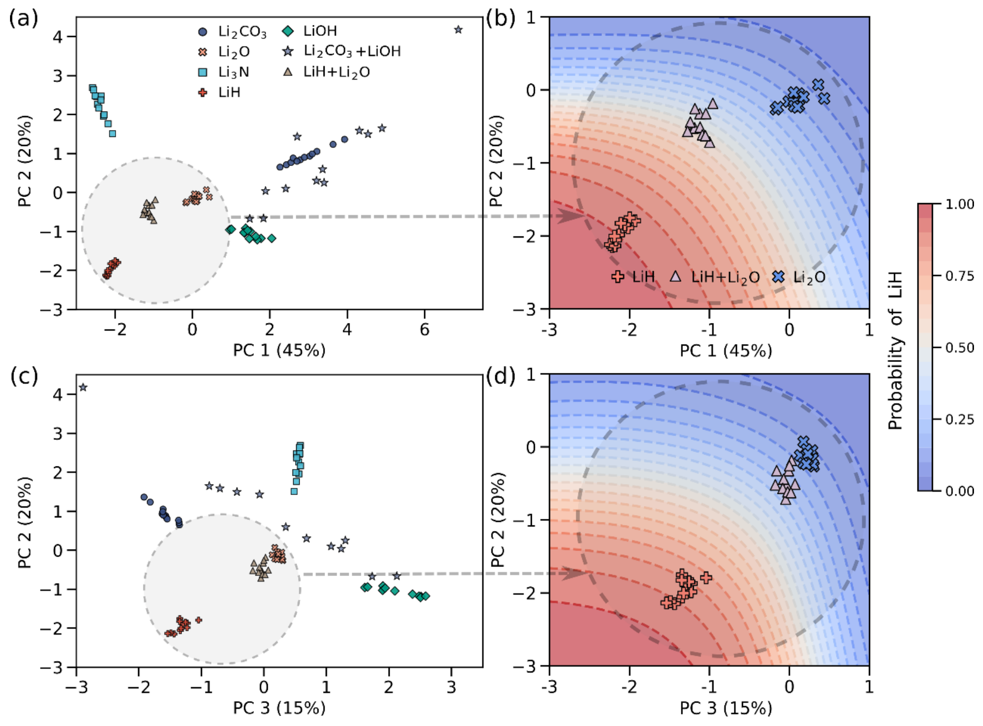


Figure 4.20: 2D scores plot of the first four PCs for the 5 pure lithium compounds and 2 mixtures, namely Li_2CO_3/LiOH and LiH/Li_2O. The PCs are numbered according to the variance (listed in parentheses) they include starting with PC1 for the highest variance. Reproduced with permission [509].

Table 4.9 reports the probabilities predicted by the LR model for the two prepared mixtures (samples No. 1–2). For these mixtures, the model can correctly predict their compositions as it yields the highest probabilities for these corresponding compounds. Before examining the predicted values in detail, it should be noted that during the measurement of SIMS spectra, the secondary ion currents are ideally proportional to the amount of a species, but it usually could not strictly reflect the quantitative composition of the measured sample due to varying ionization probabilities caused by matrix effects. Therefore, the predicted values from the model are used only as a qualitative indicator rather than a quantitative metric. For the mixture sample No. 1, a mixture of Li_2CO_3 and LiOH in 1:1 weight proportion (the molar ratio is about 1:3.1), the model predicts a probability value of 0.54 for Li_2CO_3 and 0.46 for LiOH, which matches well the position of these mixture samples in Figure 4.20: they are indeed located in the middle of pure Li_2CO_3 and LiOH compounds, explaining why the model makes such a prediction of roughly equal proportions. On the other hand, for mixture No. 2 (1:1 weight mixed from Li_2O and LiH), the model predicts a probability of 0.83 for Li_2O and only 0.14 for LiH, implying that they may contain more Li_2O. This can be supported by the fact that these mixture samples lie towards Li_2O in decision boundary plots, indicating they are more similar to Li_2O. Furthermore, it is found that the model also predicts a very small value of 0.03 for Li_3N, which is not expected in this sample. Such a prediction is presumably due to the interference from noisy peaks, and a closer examination is given in a later section. Overall, the analysis of the above results reveals that the possible compounds in a mixture can be qualitatively identified according to the predicted probability values from the LR model, though these values do not strictly correspond to their expected molar or weight ratios. Furthermore, the model can predict compounds that do not show specific peaks, namely Li_2O, which could be helpful for an accurate determination.

Table 4.9: Predicted probability for mixture and LMA samples from the LR model.

Sample No.	Compositions (expected)	Category	Remark	Composition probability				
				Li_2CO_3	Li_2O	Li_3N	LiH	LiOH
1	Li_2CO_3/LiOH	mixture	1:1 mixed	**0.54**	0.00	0.00	0.00	**0.46**
2	LiH/Li_2O	mixture	1:1 mixed	0.00	**0.83**	0.03	**0.14**	0.04
3	Li_3N	LMA	N_2-plasma treated	0.01	**0.12**	**0.83**	0.00	0.00
4	Li_2CO_3/LiOH	LMA	smaller sputter dose	**0.72**	0.00	0.02	0.00	**0.26**
5	Li_2O/LiH	LMA	double sputter dose	0.00	**0.40**	0.05	**0.55**	0.00

Identification of compositions for LMAs

Based on the analysis above, the LR model is applied to the real LMA samples (No. 3–5 in Table 4.9). Besides, to better investigate and understand the reasons for the model to make such predictions, the intensity values of the LMA ToF-SIMS spectra in the top 5 characteristic m/z peaks of the five examined pure compounds, are exhibited in Figure 4.22. For better comparison, all spectra data are scaled together using the min-max scaling method.

Sample No. 3 is a lithium metal foil that was exposed to N_2-plasma to produce Li_3N on its surface. As can be read from Table 4.9, the model predicts the highest probability value for Li_3N (0.87), which is in line with the expectation of nitride formation and can also be demonstrated by the large intensity values of these samples in the characteristic m/z peaks of Li_3N (shown in Figure 4.22). It is worth noting that the model outputs a small probability value for Li_2O (0.12), suggesting the possible presence of Li_2O. To investigate this inference, XPS measurements are performed on this sample, whose results not only confirm the formation of Li_3N but also reveal that there are indeed some oxide residuals on this sample (Figure 4.21). These are presumed to be either introduced by oxidation processes during sample transport or other contamination. Thus, the model successfully helps to identify the component that may be overlooked otherwise, and it reminds researchers to process the samples carefully to reduce possible influencing impurities.

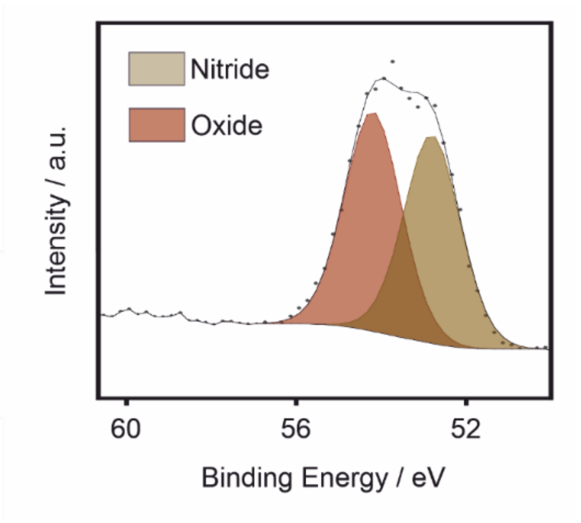


Figure 4.21: For sample No. 3, a lithium metal foil that was exposed to N_2-plasma, the presence of Li_3N and Li_2O could be confirmed by the X-ray Photo-electron Spectroscopy (XPS) in the Li 1s region. XPS measurements were conducted as reported previously after sputtering with Ar^+ with 1 kV for 3 min, 2 kV for 8 min, and 4 kV for 30 min. Reproduced with permission [509].

In the following, particular attention will be given to the other two LMA samples, as they allow discussion of potential difficulties in the interpretation of the LR results, and therefore its limitations. It should be noted that during the training process, the LR model has only seen the data of pure lithium compounds, but it is now applied to unseen samples that may contain multiple components. Before measuring their ToF-SIMS spectra, different sputter times were applied. The longer the sputtering time is, the more sample volume is ablated, and the subsequent analysis should give information on deeper parts of that sample.

For sample No. 4, only 1 frame ($= 9 \times 10^{13}$ ions/cm^2, smaller sputter dose) was sputtered to remove contaminants from the very outermost surface layer. The model correctly predicts the presence of Li_2CO_3 (0.72) and LiOH (0.26) on this sample, which is consistent with the XPS analysis of lithium metal foil, showing the existence of Li_2CO_3 and LiOH. However, a direct comparison between their results is not possible, as ToF-SIMS is more surface sensitive (1–3 upmost atom layers) than XPS (about 10 nm). Therefore, these results could indicate that Li_2CO_3 and LiOH are the dominating components on the very top of the sample surface and consequently mainly detected with ToF-SIMS. In addition, it should be noted that an unusual data distribution of this sample can be observed in Figure 4.22, as intensity values in some peaks are even higher than the one of the pure samples. For example, in the characteristic peak of Li_2CO_3 where $m/z = 59.99$ (CO_3^-), the intensity value of sample No. 4 is even ~3.5 times higher than that of the pure Li_2CO_3 sample. This phenomenon is understandable as this sample was measured after a shorter sputter time to get information on the outermost layer. Such a large value – though this signal may come from additional contamination like hydrocarbons – could dominate the model's behavior and may lead to misleading results. Hence, this sample is regarded as an outlier and to mitigate the possible negative impact, a data treatment is performed to truncate the particularly high peak values, to be specific, a maximum truncation method is adopted as one step in the preprocessing to mitigate the possible negative effects of outliers. That is, if the peak intensity exceeds the maximum value of pure substances, a truncation is performed. After this treatment, the prediction results have improved greatly, otherwise, the LR model, which was trained only on pure lithium compounds, would assign a probability of 100% Li_2CO_3, and not even a very small probability associated with LiOH.

For sample No. 5, a sputtering time of 150 seconds was applied prior to the ToF-SIMS measurements, which allows the investigation of a deeper part of the lithium metal foil. The model predicts the presence of Li_2O (0.40), and XPS analysis also indicates the presence of Li_2O mainly after longer sputtering, while going even deeper in the sample can lead to the detection of lithium metal. Additionally, the model also predicts LiH (0.55) on this sample, which is difficult to determine with XPS measurements, as it can only rule out the presence of large amounts of LiH, making the potential identification of LiH through ToF-SIMS especially interesting. To further study this, the original ToF-SIMS spectra are investigated. Considering the longer sputtering time that was applied to sample No. 5, its outermost layers should have already been removed completely before the analysis. This agrees with the lack of peaks associated with Li_2CO_3 (illustrated in Figure 4.22), explaining why the model predicts a probability value of 0.0 for this compound. On the contrary, certain signal intensity is detected in the characteristic m/z peaks of LiH, which further confirms the accuracy of the model.

Besides, the very small probability values associated with Li_3N for sample No. 4 (0.02) and No. 5 (0.05) attract attention and raise concerns about the model's prediction. Similar to the case of the mixture sample No. 2, the original peak profiles of this sample (especially characteristic peaks for Li_3N) are carefully examined to investigate whether the model has misclassified species. From the peak profiles (Figure 4.23) it is known that certain strong intensities are detected in some characteristic peaks for Li_3N, which suggests a possible presence of Li_3N on these samples, showing model's predictions are reasonable. However, these peaks seem to be very noisy, which may interfere with the model's prediction, explaining why the LR model predicts a very low probability value of Li_3N. However, it is not particularly clear where these peaks originate from. Possible reasons could be that a small portion of particles was brought on these samples when sputtering the Li_3N LMA sample, as the samples were in the analysis chamber at the same time.

Alternatively, another option is that these two LMA samples were brought into contact inadvertently. On the one hand, the short period of sputtering time on sample No. 4 was not sufficient to remove all contamination on the sample surface, which might lead to different charging of the LMA sample. This can result in a different data distribution in the measured spectrum of this sample compared to those of pure samples, for which a longer sputter time was applied. On the other hand, sample No. 5, after a longer sputtering time, has possibly reached the metallic lithium area, the signal of which may add a different intensity distribution to the measured spectra, which may interfere with the model to some degree, making an accurate prediction challenging. However, as the correlation of sputter times in ToF-SIMS and XPS is difficult and the lithium metal does not show any specific signals in ToF-SIMS analysis itself, from the available results alone it is impossible to infer whether the lithium metal is present or not. When this is the case, researchers need to examine the raw data to ensure the reliability of the model. Nevertheless, such samples can still be analyzed more easily and quickly with the help of the characteristic peaks previously identified by the model. Lastly, the importance of considering comparable sputter times and the need to reduce surface contamination should be emphasized, which are critical to ensure the accuracy of machine learning models applied to ToF-SIMS analysis.

Finally, it should also be noted once more that the LR model has only seen the data of pure lithium compounds, but it is now applied to unseen mixture samples with unusual data distribution (extrapolation), which is a considerably challenging task. A possible improvement would be adding mixture samples to the training process. However, the scarcity of mixture samples in this work hinders this realization. To overcome this problem, a data augmentation method is used to synthesize pseudo-mixture samples, with which the original problem of identifying composition is transformed as a regression problem, and a random forest regression model is fitted to predict the quantity of substance (detailed in the later section). Such a model can better predict the presence of Li_3N, which can serve as a supplement and comparison to the LR model. In addition to this, increasing the number of data samples, e.g., performing ToF-SIMS imaging to record richer information, or preparing more samples of various pure substances to enrich the database, is expected to improve models' performance and is considered as a future work.

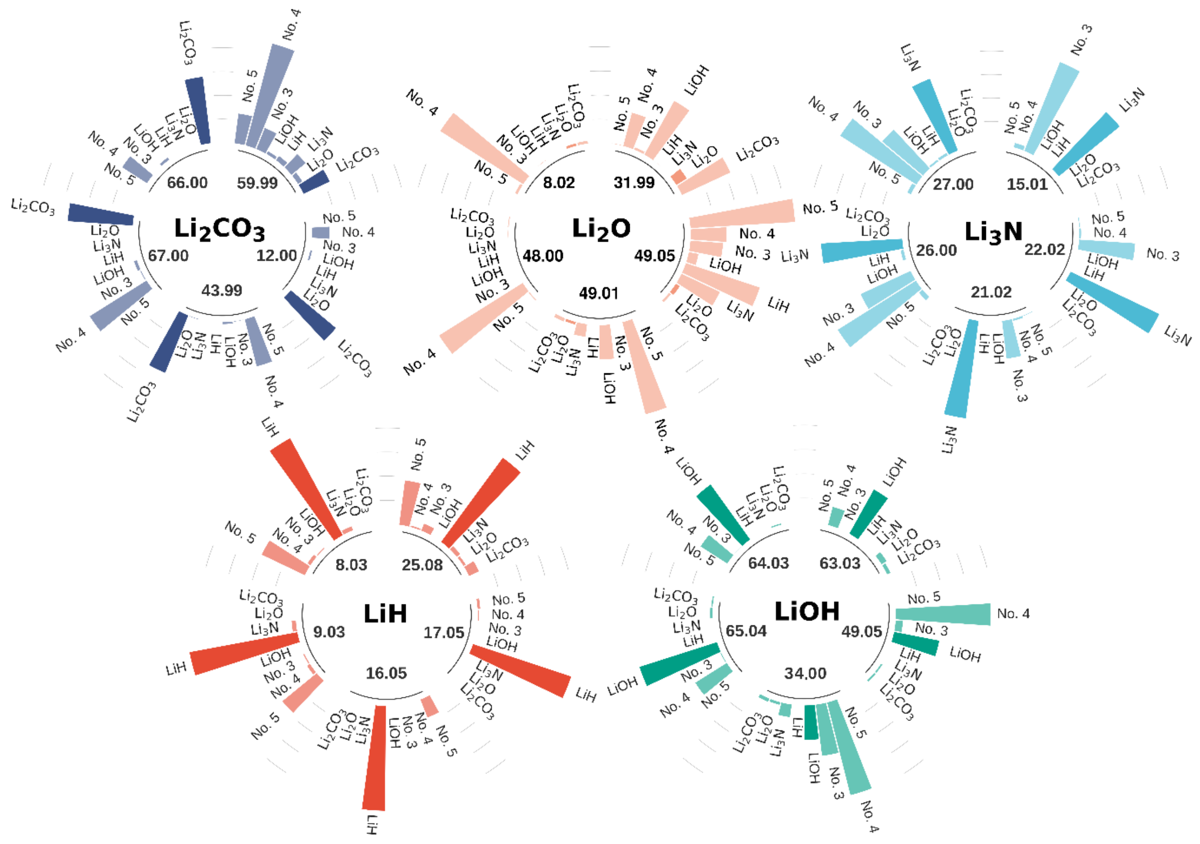


Figure 4.22: Scaled intensity values for the 5-top specific m/z peaks (labeled in each subplot) for 5 pure lithium compounds and LMAs samples in these peaks. Each circular area plot (in polar coordinate) represents one pure compound, where its corresponding intensity in each m/z peak is marked in a darker color, while other materials are in a lighter color. Reproduced with permission [509].

Analysis of predicted Li_3N on the mixture and LMA samples

The LR model predicts a very small possibility value for Li_3N for samples No. 2, No. 4, and No. 5. To investigate whether the model has misclassified species on these samples, their peak profiles are carefully inspected and the 15-top Specific m/z Peaks (SP) of Li_3N are exhibited in Figure 4.23. It can be observed that samples No. 2 and No. 5, do show intensities in some m/z peaks of Li_3N (but not in all SP), explaining the reason why the LR model suggests smaller amounts of Li_3N on these samples. It can also be seen that the peak intensities of sample No. 5 are generally higher than those of sample No. 2, explaining why the model yields slightly higher values for Li_3N on sample No. 5 than on No. 2. On the other hand, sample No. 4 is inferred to contain much more Li_3N than the other two based on the SP plot, however, the LR model only predicts a small possibility value (0.02) for Li_3N. The reason why the model only yields a small probability value is inferred to be caused by the unusual data distribution (the excessive peak intensity) of this sample compared to the training set, leading to the interference. Such deviations in its data distribution, while large, are common occurrences in data analysis, in particular in treating outliers. Meanwhile, such kind of predictions can be expected, since the LR model is trained on a limited number of pure substances and generalized to new unseen mixture samples with large differences in data distribution.

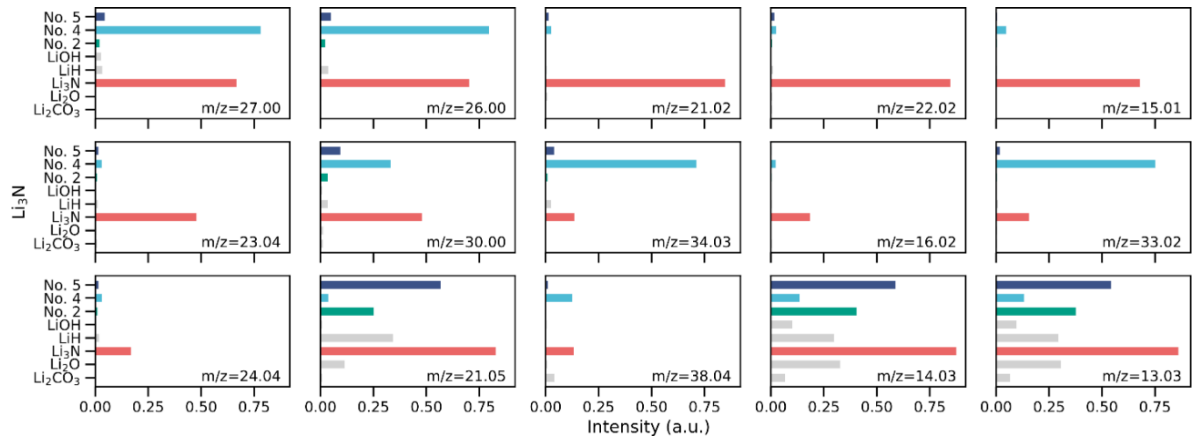


Figure 4.23: 15-top specific m/z peaks of Li_3N (colored in red). For better comparison, the other four pure lithium compounds (colored in gray), mixture sample No. 2 (LiH+Li_2O, colored in green), LMA sample No. 4 (colored in light blue), and LMA sample No. 5 (colored in dark blue) are also exhibited. Reproduced with permission [509].

4.3.4 Other methods and perspectives

Decomposition of the mixture and LMA samples

In this section, the Non-negative Matrix Factorization (NMF) method is applied to the mixture and LMA samples to analyze their compositions. Basically, the NMF method attempts to separate the independent original signal from the mixed signal source. NMF works by first assembling a matrix $\boldsymbol{A}$, which contains samples in its rows and variables (spectral channels) in its columns, and then factorizing it approximately into non-negative matrices $\boldsymbol{W}$ and $\boldsymbol{H}$ using a multiplicative algorithm, i.e., $\boldsymbol{A} \approx \boldsymbol{W}\boldsymbol{H}$. $\boldsymbol{H}$ represents the characteristic spectra of a specific number of "pure compounds" presented in the samples and $\boldsymbol{W}$ represents their distribution maps, which can be interpreted as the "concentrations" of these compounds in each observation. An important parameter for this method is the number of components n to be decomposed. Since it is not known in advance how many independent components a mixture sample consists of, a simple heuristic, the "elbow" method is used to determine the optimal number in this work: a relationship diagram is plotted with n ranging from 2 to 5 (on the horizontal axis) against the corresponding reconstruction error (on the vertical axis), from which the "elbow" point with the largest rate of error change (the point where the rate of reduction drops significantly) is selected as the best number.

For the two mixture samples No. 1–2, decomposition results obtained from the NMF method are shown in Figure 4.24. It can be deduced that both mixture samples are mainly composed of two independent components according to the relationship plots (n vs. reconstruction error), as a sharp turn can be observed at the elbow point where $n = 2$. This is consistent with the fact that the two mixtures were made by mixing two pure substances respectively. For each mixture sample, its recovered peak profiles (decomposed spectra of “pure compounds” according to matrix $\boldsymbol{H}$) sub-scripted with (i) and (ii), together with their best-matching pure lithium compounds, are also exhibited for better comparison. The matching similarity is measured by calculating the Pearson correlation coefficient p (a value between -1 and 1, where a larger value reflects a stronger positive relationship) between the peak profiles [517]. It can be seen that except for Li_2O, other reconstructed profiles have a high degree of agreement (> 0.98) with those of the pure substance. In addition, their relative “concentrations” can be quantified by evaluating the NMF intensities (distribution maps in matrix $\boldsymbol{W}$) [518, 519]. For sample No. 1, it is found to consist of 75% Li_2O and 25% LiH while sample No. 2 consists of 67% Li_2CO_3 and 33% LiOH. From these results, it can be concluded that the NMF method performs well in decomposing the artificially manufactured mixture samples and can be used to assist researchers in accurately identifying the composition of these samples.

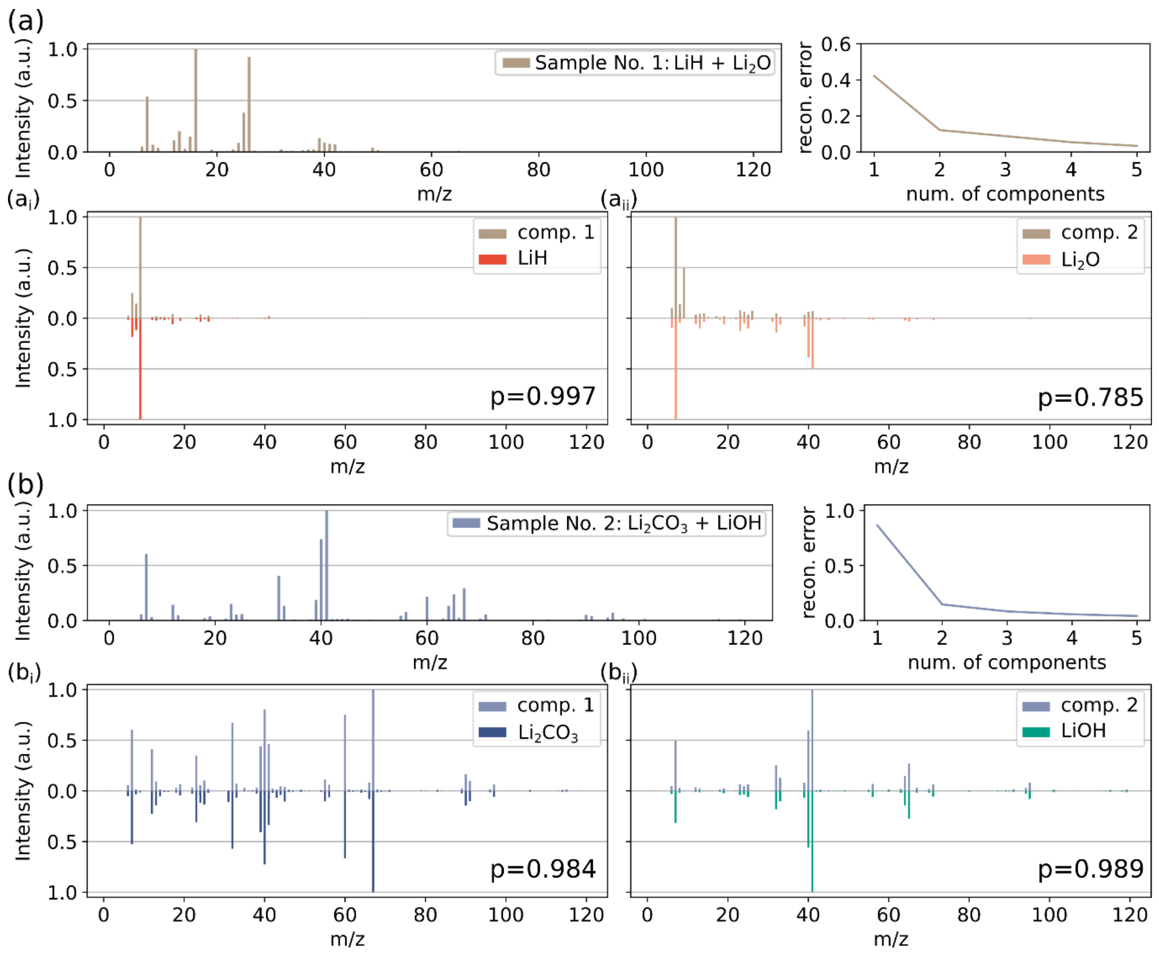


Figure 4.24: Decomposition results obtained from the NMF method for the two mixture samples (No. 1 and No. 2) are shown in (a) and (b), respectively. In each sub-figure, the peak profile of the mixture sample, the change of reconstruction error with respect to the number of components, and the recovered individual component profiles (i)–(ii) along with the best matching peak profiles of pure compounds (measured in Pearson correlation coefficient, abbreviated as p and marked in the lower right corner of each decomposed component plot), are exhibited. The peaks are normalized to the maximum of their corresponding profile for better visualization and comparison. Reproduced with permission [509].

Similarly, the NMF method is also applied to LMA samples No. 3–5, and their results are shown in Figure 4.25. The case of real LMA samples is more complicated than that of artificially mixed samples. For sample No. 3, it is inferred to be composed of two components, however, both components are inferred to be Li_3N (i.e., 100%) as they are all best matched with the pure Li_3N compound. Pearson correlation coefficient is used to measure the similarity between profiles, generally, a higher value implies a better match. From Figure 4.25a it is observed that component 1 with a higher p-value (0.753) is indeed more visually similar to Li_3N than component 2 ($p = 0.607$). Sample No. 4 is estimated to contain four components, three of them (components 1–3) are found most similar to

LiOH (83%, in line with expectations), and the 4-th component is closest to Li_3N (17%) but with lower p-value (0.438). For those three components matched with LiOH, it can be observed that the match of the spectra profile with LiOH gradually becomes poorer as the p-value decreases, and for the third component ($p = 0.411$), there are many mismatched peaks and only some characteristic peaks are well aligned. For sample No. 5, it is inferred to consist of three components: LiH (67%), Li_2O (25%), and Li_3N (8%). Among them, the recovered peak profiles, which are inferred to be LiH ($p = 0.891$) and Li_2O ($p = 0.660$), match the expected compositions of this LMA sample. The obtained 3rd component is found to be most like Li_3N, however, its relatively low p-value (0.415) implies it may be an inaccurate result. This can also be noted from the visual comparison between their peak profiles, as only a limited number of the characteristic peaks are properly matched. Taking into account the results of the three LMA samples, a threshold value of p-value is needed to filter out the poorly matched components, and an empirical value of 0.5 is picked for this study.

In summary, the results of the five analyzed samples have demonstrated the limited effectiveness of adopting NMF in the analysis of mixtures, as it may miss components or give misleading results. For the homogeneously blended, artificially fabricated samples, this method allows accurate separation of the original independent components. This is reasonable because these mixtures were actually made by milling the pure samples and their profile nature should be closer to those of the pure substances, resulting in better decomposition performance. However, for real LMA samples, this method is not always effective, which can lead to failure in identifying some constituents or even giving wrong outcomes. In addition, a higher level of manual intervention is still required, such as visual inspection or setting threshold values based on the researcher's experience for filtering possibly misleading results. Besides, it is worth pointing out that the number of data samples in this study is very limited, and the increase in the amount of data is expected to improve the accuracy of the decomposition result.

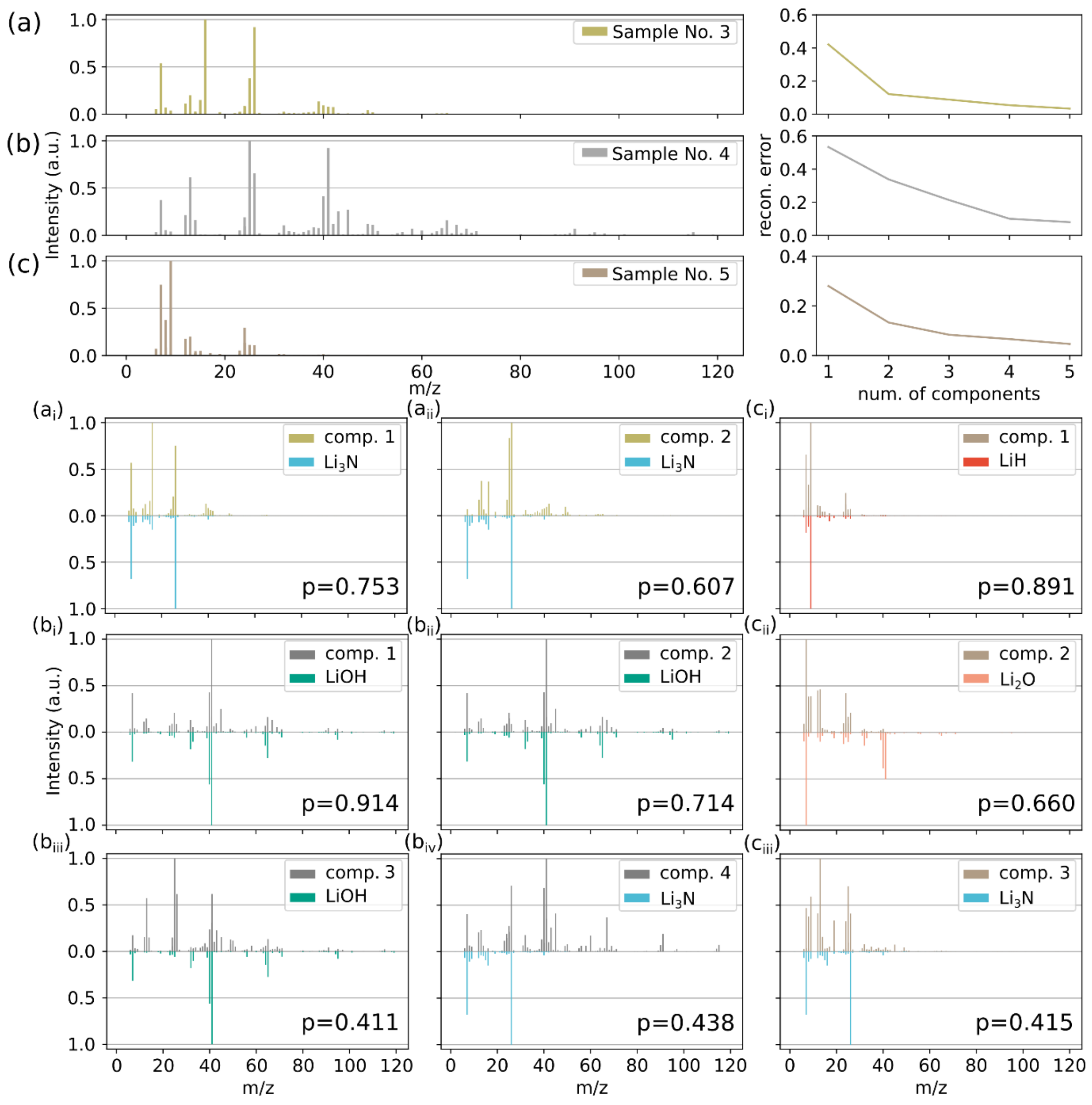


Figure 4.25: Decomposition results obtained from the Non-negative Matrix Factorization (NMF) method for the real LMA samples No. 3–5 are shown in (a)–(c), respectively. In each sub-figure, the peak profile of the mixture sample, the change of reconstruction error with respect to the number of components, and the recovered individual component profiles sub-scripted with the symbol (i)–(iv) along with the best matching pure compounds (measured in Pearson correlation coefficient, abbreviated as p and marked in the lower right corner of each decomposed component plot), are exhibited. The peaks are normalized to the maximum of their corresponding profile for better visualization and comparison. Reproduced with permission [509].

Predicting compositions using regression models

It is another option to have models trained also on mixture data and treat the task of identifying the compositions of the mixture and LMA samples as a multi-label classification problem. However, the scarce number of mixture samples in this work hinders this realization. To alleviate this, the discussed problem is transformed into a regression task: a regression model can be trained to predict the amount of different pure substances in a sample, enabling the identification of its compositions according to the predicted quantity values. To make use of the limited amount of data, a data augmentation approach is used: new pseudo-mixture samples are made by randomly selecting two peak profiles of different substances and then the two are mixed by multiplying random weights for each (the two weights are seen as the quantities of substances in a mixture, and they sum to 1). The task of the regression model is to predict these weights (i.e., the quantity of the corresponding substances) based on these synthetic peak profile data. To achieve satisfying results, approximately 10k pseudo-mixture samples are generated.

To evaluate the performance of the model, the coefficient of determination R^2 is adopted (larger value, better result). The studied machine learning regression models are: linear regression [520, 521], Support Vector machine Regressor (SVR) with linear or Radial Basis Function (RBF) kernel, decision tree, and its advanced variants: Random Forest (RF), Adaptive Boosting (AdaBoost) method, gradient boost machine (all with decision tree as the base estimator) [69], Gaussian Process Regression (GPR) [522, 523], and a shallow neural network. The studied problem is a multi-output task (5 outputs for 5 pure substances), and if the model does not natively support multi-target regression, a simple strategy is adopted, i.e., training multiple models, each fitting one target. The whole data set is divided into 60/20/20% as training, validation, and test sets. For each model, it is tried ten times on the training set with different hyperparameters, and the validation set is used to determine the most appropriate one. Afterward, the model is trained with the found best hyperparameters on a data set assembled by the training and validation sets, and the final performance of the model is evaluated on the test set. Among the tested models, the RF model (number of trees = 200) achieves the best performance (R^2 = 0.999).

This model is then applied to the mixture and LMA samples for composition identification and their results are summarized in Table 4.10. According to the evaluation criteria described in Section 4.3.3, the regression model shows good results on these samples. Compared with the LR method that relies on the classification probabilities to determine the compositions, it is found that both methods yield essentially similar conclusions despite differences in the specific values. It should be emphasized that the two sets of results are only qualitatively compared since the probability values obtained from the classification model are not strictly equivalent to the amounts of the substances. Moreover, it is also worth noting that for sample No. 4 the data are also pre-treated with the maximum truncation method, and the regression method predicts a small value (0.1) for the presence of Li_3N on it (where the LR model predicts an almost negligible value 0.02). The most likely reason for the presence of Li_3N on sample No. 4 is presumed to be residual contaminants on the sample surface and it is discussed in the main text. As can be concluded from the above results, treating the original problem as a regression problem allows it to better identify some compositions on a mixture sample, although certain data enhancement and longer time costs of the fitting model are needed. These results can be a good complement to the original analysis, opening a new perspective to analyze the problem of composition identification.

Table 4.10: Predicted probability for mixture and LMA samples from the RF model.

Sample No.	Compositions (expected)	Category	Remark	Composition probability				
				Li_2CO_3	Li_2O	Li_3N	LiH	LiOH
1	Li_2CO_3/LiOH	mixture	1:1 mixed	**0.65**	0.00	0.00	0.00	**0.35**
2	LiH/Li_2O	mixture	1:1 mixed	0.00	**0.73**	0.00	**0.27**	0.00
3	Li_3N	LMA	N_2-plasma treated	0.04	**0.20**	**0.74**	0.00	0.02
4	Li_2CO_3/LiOH	LMA	smaller sputter dose	**0.55**	0.01	0.10	0.00	**0.34**
5	Li_2O/LiH	LMA	double sputter dose	0.01	**0.50**	0.00	**0.49**	0.00

4.3.5 Conclusion

In summary, the LR model performs satisfactorily on the three tested LMA samples despite the challenge of extrapolation. With the help of the characteristic peaks obtained from the model and the predicted probability associated with the different compounds, the compositions of the mixture and LMA samples can be identified qualitatively in a faster and simplified way. Besides, it is worth noting that detecting and handling the outlier samples are necessary steps to offer a more robust result. Hence, a more elaborate measuring procedure would need to be designed to ensure as much as possible a consistent distribution of data across the sample to mitigate possible negative effects. Moreover, depth profiles can be measured to investigate probability change as a function of sputter time utilizing the high sensitivity and lateral resolution of ToF-SIMS. This could be helpful in identifying any artifact introduced by sputtering and assist in making more confident interpretations about oxidation, surface segregation, and other related activities such as the matrix effects [524, 525].

Chapter 5

Conclusion and outlook

5.1 Summary and perspective on materials research

In this work, a comprehensive and structured overview of applying machine learning in the field of materials science is presented. It systematically distills and explains the essential core principles, methodologies, and theoretical foundations necessary for comprehending and engaging with machine learning paradigms in materials research. These applications have become increasingly prevalent in recent years due to the unparalleled successes that machine learning has achieved in various scientific and technological domains.

For the three primary components of materials research, namely, experimentation, simulation, and characterization, this work shows example studies to demonstrate the advantages and effectiveness of using machine learning methods to solve practical problems during the research process. The presented studies cover typical application scenarios and cover a wide range of research areas, scales, and data types. For application to experiments, Bayesian optimization is utilized to design experiments for synthesizing an oxide electrolyte in a solid-state battery. The goal is to minimize the required number of attempts, resulting in a reduction of overall time. This method has proven to be helpful in assisting researchers in making decisions and is approximately 2–3 times faster than traditional methods. For application to simulation, a neural network model, specifically a variational autoencoder, is used to automate the extraction of structural information of digitally generated porous microstructure. This structural information can then be represented in low-dimensional variables, enabling the development of fast predictive models for building process-structure-property relationships, which is a critical topic in materials studies. For application to characterization, using the one-dimensional data of time-of-flight secondary ion mass spectrometry as an example, this work explores the effectiveness of different machine models and makes comparisons between the results.

Not only a fast classification and identification model has been built, but interpretable chemical information can be retrieved based on the feature importance from the model.

Throughout the course of this work, many discussions were held with collaborators during the realization and completion of these studies. In addition, to better understand the current status and opinions of applying machine learning methods in materials research, surveys and workshops were conducted with project partners, from which many reflections and views were gained and they are discussed in the following.

Individual data workflows and missing data standards

Data workflows within the materials field exhibit considerable variability depending on factors such as specific research focus, data acquisition techniques, and individual researcher preferences. Although materials researchers can generate extensive knowledge about the relationships between processing, structure, properties, and performance of materials, the differing methods of data generation, analysis, and dissemination present significant challenges to the reuse and re-purposing of others' data. Through surveys with materials researchers at academic institutions or in industry [526] and experience from this work, each individual researcher can make different choices among characterization techniques, simulation tools, data management platforms, and data analysis methods. This fragmentation impedes the centralization and standardization of materials data. Therefore, fostering cross-disciplinary collaboration and establishing a comprehensive, centralized data clearinghouse will be vital for advancing materials research, which needs to be designed to organize and store diverse data more effectively.

The current landscape of materials data appears as a disjointed mosaic of smaller databases, each catering to specific sub-disciplines. The majority of existing material data resources are currently geared towards "low-throughput" human consumption. In contrast, modern data analytics techniques require systematic access to extensive data sets, whereas the current state of materials data is fundamentally incompatible with cutting-edge computational methods for gaining insights. Currently, many resources utilize idiosyncratic data formats and the absence of a standardized data format is a significant obstacle to improving the materials data landscape. This lack of standardization can impede reuse and analytics due to interpretability challenges posed by the varied data formats. Moreover, the absence of data standards creates additional challenges when attempting to extract valuable information from extensive material data resources. In order to overcome this issue, it is imperative to establish adaptable, standardized, and machine-readable data protocols, which will facilitate organized data management, distribution, analysis, and sharing. Achieving broad agreement on data standards among various stakeholders is crucial and necessitates the cooperation of multiple working groups spanning the globe.

Diversity of stakeholders and research areas

As a vast and multidisciplinary field, materials science flourishes through intricate collaborations between data producers (researchers from universities, government laboratories, and industry), funding agencies, instrument manufacturers, software developers, and distributors of research results (often represented by journal publishers). While these stakeholders are important to materials research, their motives for making materials data widely available are not always aligned. For instance, researchers may opt to withhold noteworthy findings to postpone potential publication by others, and unfavorable outcomes might be completely concealed. Industrial researchers often need to protect their most valuable results as trade secrets, while equipment manufacturers may develop proprietary data formats to differentiate their tools through software. Publishers may provide limited or paid databases, driven by a financial incentive to control access to publications and data. In addition, the establishment of a uniform framework to standardize all the data is challenging due to variations in material systems and scales.

Lack of incentives and questioning reliability

Currently, materials researchers have little motivation to share their data. It is uncertain whether making research data widely available in the materials community will lead to benefits such as increased academic impact and citations, improved funding prospects, or enhanced opportunities for professional advancement and promotion. Some countries are starting to require a data management plan for the funded research. Future efforts may focus on encouraging data sharing by funding agencies and journal publishers, either through rewarding researchers for community data contributions or by prescribing data management practices.

Beyond the absence of incentives for data sharing, the materials community also lacks a compelling motivator for a structured data access and advanced analytics culture. Many materials researchers persist in utilizing conventional “small data” generation and analysis methodologies, neglecting the advantages that can be gained from applying large-scale data analysis techniques in their research. In addition, a number of machine learning methods remain suspect due to their “black box” nature, i.e., lacking clear explanatory properties. Researchers tend to reject methods that they find difficult to interpret. Indeed, in many cases, the predicted results from machine learning models still need to be verified experimentally and the underlying physical/chemical principles need to be investigated in detail. The issue of interpretability is also a hot research topic in the field of machine learning. Solving this problem requires not only the advancement of the field but also more cross-disciplinary scholars to combine these techniques to promote practical applications. Besides, there are still many misconceptions surrounding machine learning and related concepts. Therefore, it is crucial to increase awareness and provide better training for materials scientists. This will enable them to appreciate the significance of standardized data and instruct them on how to utilize data-driven research methodologies for their own research data. Such kind of training should include practical guidelines and examples for data acquisition, refinement, accessibility, and mastery of relevant tools and competencies.

5.2 Comment and future direction

In the realm of data-driven materials science, many barriers have been identified which hinder modern materials research, however, there is a positive shift in the landscape of this field. Open-access content is on the rise, with a growing number of publishers embracing open-access models and releasing papers under free licenses that enable users to access content and data at no cost. Governments worldwide are working to improve accessibility to research data and to develop a standardized approach for open data sharing to facilitate seamless collaboration. Funding agencies are recognizing the significance of meta-data and data provenance in materials research while industrial stakeholders are acknowledging the urgency of open manufacturing data and unified standards.

Notable projects are emerging to build a data management platform for organizing research data, as well as to develop unified workflow tools capable of integrating diverse software to streamline research processes. Repositories are also being developed to support structured data formats, resulting in varying levels of accessibility. The in-house implemented scientific data management framework Kadi4Mat [527], and the standardized workflow tool KadiStudio [528], which are under active development by the group, will be powerful tools for helping researchers in this community better manage data in an FAIR manner [529] and streamline daily research work. The generated data in this work and adopted machine learning methods, should be better structured and integrated into these tools as examples for reference by relevant parties who are interested.

Furthermore, it is recommended to increase awareness of machine learning applications in materials research. This can be achieved through the organization of workshops that illustrate fundamental concepts and offer practical guidance, facilitating a swift comprehension of the topic for researchers. In order to reduce entry barriers and facilitate the rapid development of machine learning models to solve practical problems in materials research, low-code platforms should also be developed. An example of this is the interactive and visual dashboard in the KadiAI framework [328], which is being developed by the working group actively. Furthermore, it is imperative to encourage the involvement of more interdisciplinary scholars in this field, as it will increase practical illustrations, widespread interest, and broader cooperation.

The materials science community currently lags behind other scientific disciplines in the integration and application of large-scale data-driven methods, which presents both challenges and opportunities. Fortunately, there is a noticeable and improving trend toward remedying this situation. Ongoing and related initiatives are expected to strengthen materials research by building and expanding access to large, well-organized data sets, where text-mining techniques are expected to be very helpful. Additionally, increasing interdisciplinary collaboration can broaden the applications of machine learning in materials science, ultimately benefiting the field of materials research. Time serves as the fertile ground for growth and harvest, let us witness the dawn of a new era in this field.

Appendices

A. Analysis of latent variables of porous membranes

In Section 4.2 of this work, microstructures of porous membranes have been fed into the encoder of the Variational AutoEncoder (VAE) model to be projected into low-dimensional latent space. In this way, microstructures are represented as latent variables, which are then processed by the Principal Component Analysis (PCA) to further refine the information and to facilitate visualization. The third Principal Component (PC3) is found to mainly correlate with information on porosity, while PC1 and PC2 together represent information on directionality. A detailed analysis is given in the following paragraphs.

The main reason for performing PCA on the low-dimensional latent variables is that in the original latent space, the most interested structural information is found to be scattered across multiple dimensions and no single dimension can be obviously well disentangled. Therefore, PCA is employed to further extract and integrate the most important information. Figure A.1a shows the cumulative explained variance ratio from the PCA result and it can be seen that to contain over 95% of the original information, all the first 228 principal components need to be considered. This suggests that the information compressed by the VAE model in the latent space is already very condensed and a further compression may not work well.

The pairwise relationships between the first 6 principal components to each other are exhibited in Figure A.2 where each data point represents one microstructure and is colored according to its stretching direction. The subplots along the diagonal exhibit distributions of the stretching directions in the corresponding principal components. It can be clearly observed that the data points can be well distinguished (in terms of stretching direction) by PC1 and PC2 and the geometry of their distribution is intuitively consistent. Certain clustering can also be observed in some other principal components, but it is not as obvious as that in PC1/2. For many principal components, it is neither easy to judge whether they can capture directional information well, nor to explain their spe-

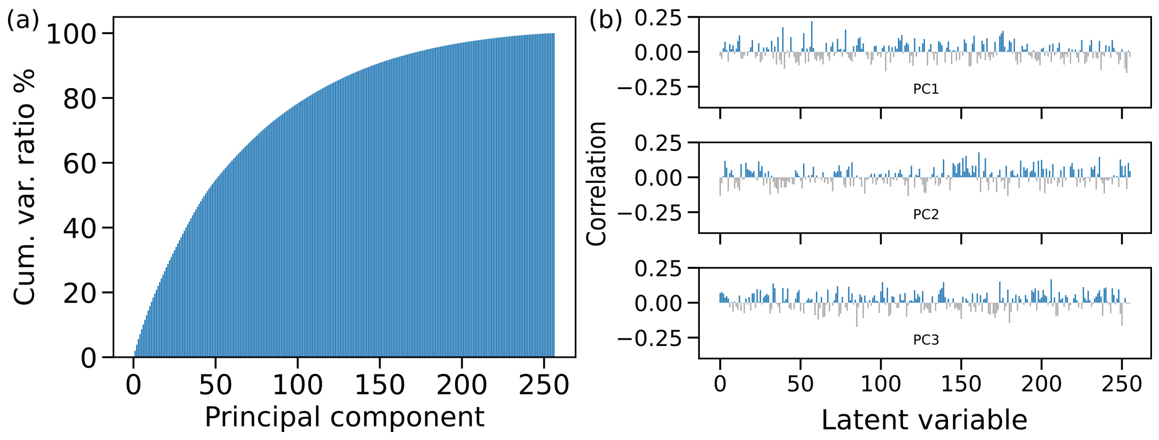


Figure A.1: Result of principal component analysis: (a) shows the cumulative explained variance ratio in all the 256 *principal components; (b) depicts the correlation loadings of the top three principal components. Reproduced with permission [499].*

cific physical meanings. Therefore, PC1 and PC2 are selected together to represent the information on the stretching direction in the microstructure. The analysis procedure of microstructural information in PC3 is similar and it is found to be mainly related to porosity/number of Voronoi points.

The correlation loadings of PC1–3 are plotted in Figure A.1b. It can be found that all these three principal components are related to multiple dimensions of the latent variables, which further indicates the original structural information is spread over multiple dimensions in the latent space. Therefore, using some dimensions of latent variables alone cannot disentangle the microstructural information well. The top 3 dimensions of latent variables that are most relevant to PC1 and PC2, respectively, are further selected and their pairwise relationships are drawn in Figure A.3. It can be seen that even choosing the dimensions that are mostly related to the directional information, microstructures still can not be well clustered and separated according to their stretching directions.

Based on the above analysis, PCA is required to be performed on the latent variables in the latent space to refine the microstructural information of interest. Apparently, microstructural information is not well disentangled in the low-dimensional latent space learned by the neural network model trained in an unsupervised way. During the training process, the neural network model is not set to learn any specific property targets rather than reconstructing the inputs and making the latent variable obey a normal distribution. As a result, it is difficult to control how the model should learn the microstructural information and assign it to the latent variables in a well-structured manner. The question of how to make the model disentangle the information effectively for building a well-structured latent space is well worth exploring. Some improvements to the basic VAE models, such as β-VAE [144], infoVAE [530] and other variants, may be helpful.

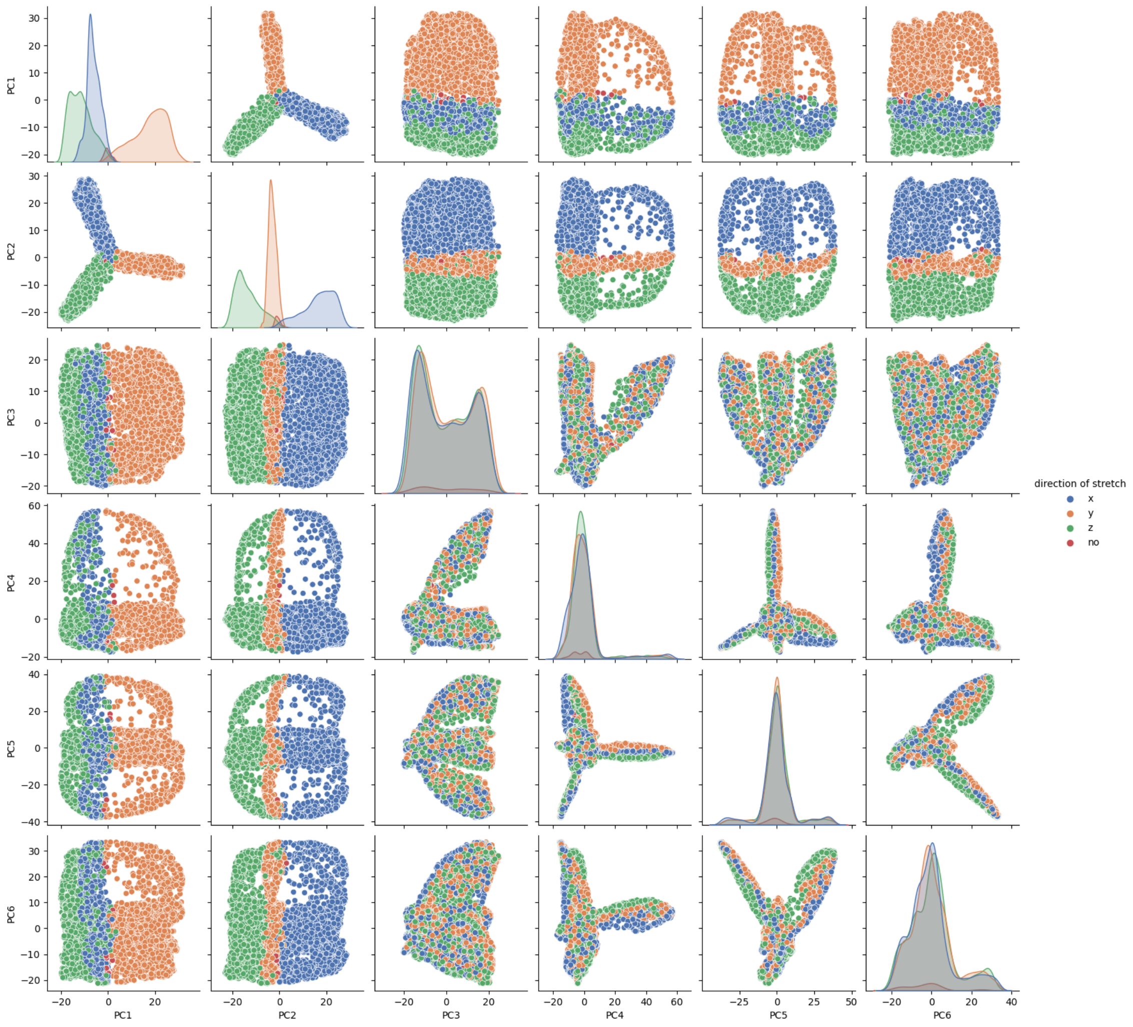


Figure A.2: Pairwise distributions of generated microstructures in the top 6 principal component space. Microstructures are colored according to their stretching directions. Reproduced with permission [499].

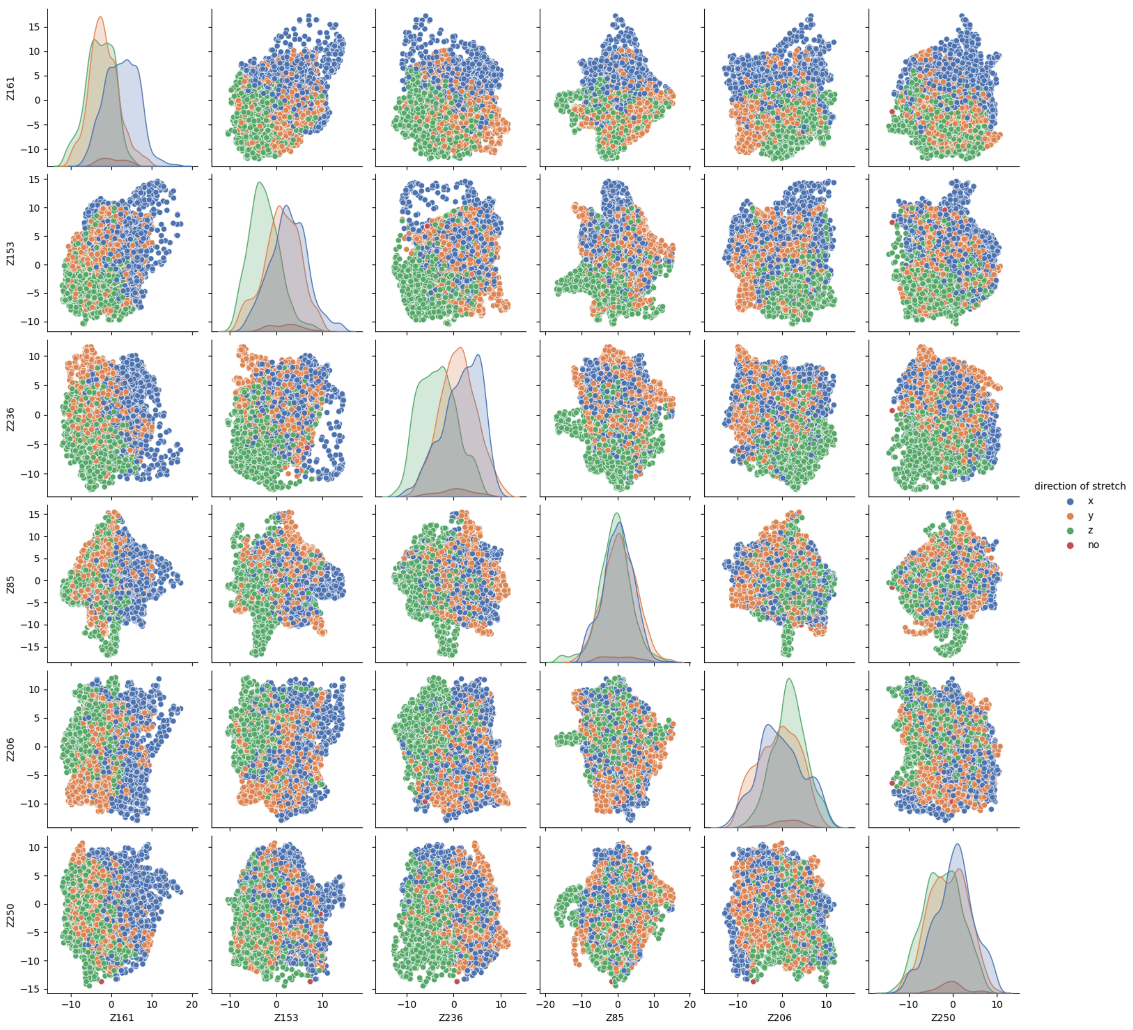


Figure A.3: Pairwise distributions of generated microstructures in the dimensions of latent space that are most closely related to PC1 and PC2, which represent directional information in microstructures, and are colored according to their stretching directions. Reproduced with permission [499].

B. Calculation of tortuosity

In Section 4.2 of this work, the tortuosity of the generated porous media is calculated by numerically solving a Laplace's equation in a Cartesian grid with the finite difference method [531]. As a special case of the Poisson's equation, the Laplace's equation is a homogeneous, second-order partial differential equation, which acts as a mathematical model to provide the common basis for describing different physical relationships:

$$\boldsymbol{\nabla} \cdot (\lambda \boldsymbol{\nabla} \Phi) = 0 \tag{1}$$

where λ stands for the general conductivity and Φ is the potential function to be interpreted. Under the assumption of source-free and turbulence-free as well as stationary potential fields, many physical quantities can be described, for example, heat conduction processes, diffusivity, and electrical conductivity. To determine the tortuosity, the analogy between the physical processes is used and, within the scope of this work, the electrical conduction is considered as the physical process to be modeled. For this purpose, a potential difference of 1 V is applied across two edges of the porous microstructure with the distance L in such a way that a direction of flow through the pore space is specified. From the following differential equation

$$\boldsymbol{\nabla} \cdot (\sigma_\mathrm{P} \boldsymbol{\nabla} \Phi) = 0, \tag{2}$$

the potential field can then be determined together with the known intrinsic conductivity σ_P, whereby a homogeneous Neumann boundary condition is defined as a further boundary condition on the surface between the pore and structure space, which prevents the flow normal to the wall. The result of the local gradient of the potential field ($\boldsymbol{\nabla}\Phi$) is the electric current I through the pore space. The effective resistance R, which leads to the effective conductivity σ_eff, follows the Ohm's law ($U = RI$)

$$\sigma_\mathrm{eff} = \frac{I}{\Delta\Phi}\frac{L}{A} \tag{3}$$

where L is the direct length of the transport path and A is the cross-sectional area. From the effective conductivity σ_eff and the ideal conductivity σ_P, in combination with the porosity ε, the tortuosity τ can finally be determined as follows:

$$\tau = \varepsilon \frac{\sigma_\mathrm{P}}{\sigma_\mathrm{eff}}. \tag{4}$$

List of symbols

x	Variable, usually a scalar feature in feature vector $\boldsymbol{x}$
y	Variable, usually a scalar label
z	Variable, usually a scalar intermediate output
C	Scalar, a hyperparamter for the number of classes
D	Scalar, a hyperparamter for the spatial dimension
K	Scalar, a hyperparamter for the number of clusters or parts
L	Scalar, a hyperparamter for the number of layers in neural networks
M	Scalar, a hyperparamter for the number of features
N	Scalar, a hyperparamter for the number of samples
T	Scalar, a hyperparamter for the length of sequence
$\boldsymbol{x} \in \mathbb{R}^D$	D-dimensional column vector in real space, usually an input sample
$[x_1, \cdots, x_D]$	D-dimensional row vector with subscript representing the dimension
$\mathbb{R}^D$	D-dimensional real space
$\boldsymbol{x}^\intercal$	Transpose of vector $\boldsymbol{x}$
$\boldsymbol{A} \in \mathbb{R}^{K \times D}$	Matrix of size $K \times D$
$\boldsymbol{X}$	Matrix, usually input samples
$\mathbf{0}, \mathbf{1}$	Vector of zeros/ones
$\boldsymbol{I}$	Identity matrix
L_1	Taxicab norm
L_2	Euclidean norm
L_p	p-norm
$\mathcal{D}$ or $\{(\boldsymbol{x}^{(n)}, y^{(n)})\}_{n=1}^N$	Set of data with superscript representing the index among N samples
$\mathcal{S}$	Subset of a given set
$\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$	Multivariate normal (Gaussian) distribution with mean values $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$
$p(\boldsymbol{x})$	Probability distribution of variable $\boldsymbol{x}$
$p(y \mid \boldsymbol{x})$	Conditional probability distribution of y given $\boldsymbol{x}$
$\mathbb{E}_{\boldsymbol{x} \sim p(\boldsymbol{x})}[f(\boldsymbol{x})]$	Expectation of function $f(\boldsymbol{x})$ where $\boldsymbol{x}$ follows $p(\boldsymbol{x})$
Φ	Cumulative distribution function

ϕ	Probability density function
$\boldsymbol{\theta}$	Parameters of the model
$\boldsymbol{w}$	Weights vector (usually for linear model)
b	Bias (usually for linear model)
Γ	Lagrangian dual function
λ	Lagrange's coefficient or strength of regularization
β	Balance factor of precision and recall
γ	Margin in support vector machine or discount factor in reinforcement learning
a	Action in reinforcement learning
s	State in reinforcement learning
r	Reward in reinforcement learning
π	Strategy in reinforcement learning
$\mathcal{A}$	Set of actions
$\mathcal{E}$	Set of edges
$\mathcal{L}$	Loss function
$\mathcal{R}$	Expected risk
$\mathcal{V}$	Set of nodes
$\mathcal{X}$	Input space
$\mathcal{Y}$	Output space
$\mathcal{Z}$	Implicit structure space
$\mathcal{F}$	Set of function
$\mathcal{GP}$	Gaussian process
$\mathscr{N}$	Neighbors of an object
x, y, z	x/y/z-coordinate
μ	Mean value
σ^2	Variance
e	Stretching factor
r	Stanchion radius
C	Inverse of regularization strength
p	Pearson correlation coefficient
λ	General conductivity
Φ	Potential function
σ_{P}	Intrinsic conductivity
σ_{eff}	Effective conductivity
I	Electric current
R	Effective resistance

U	Electric potential difference
L	Direct length of the transport path
A	Cross-section area
ε	Porosity
τ	Tortuosity

List of abbreviations

k-NN	k-Nearest Neighbor
ACSF	Atom-Centered Symmetry Functions
AdaBoost	Adaptive Boosting
AEs	AutoEncoders
AI	Artificial Intelligence
ANN	Artificial Neural Network
API	Application Programming Interface
ASSBs	All-Solid-State Batteries
AUC	Area Under Curve
BGD	Batch Gradient Descent
BO	Bayesian Optimization
CAP	Credit Assignment Problem
CART	Classification And Regression Tree
CDF	Cumulative Distribution Function
CFD	Computational Fluid Dynamics
CLIQUE	CLustering In QUEst
CNN	Convolutional Neural Network
CV	Computer Vision or Cross-Validation

DBSCAN	Density-Based Spatial Clustering of Applications with Noise
DCGAN	Deep Convolutional Generative Adversarial Networks
DFT	Density Functional Theory
DL	Deep Learning
DOS	Density of State
DRL	Deep Reinforcement Learning
DT	Decision Tree
EBSD	Electron BackScatter Diffraction
ECFP	Extended-Connectivity FingerPrint
EDX	Energy Dispersive X-ray
EI	Expected Improvement
ELBO	Evidence Lower BOund
ELU	Exponential Linear Unit
ERM	Empirical Risk Minimization
ESRF	European Synchrotron Radiation Facility
FEM	Finite Element Method
FN	False Negative
FP	False Positive
GAN	Generative Adversarial Networks
GBDT	Gradient Boosting Decision Tree
GCN	Graph Convolutional Network
GELU	Gaussian Error Linear Unit
GMM	Gaussian Mixture Model
GNN	Graph Neural Network
GP	Gaussian Process

GPR	Gaussian Process Regression
GPR-BO	Gaussian Process Regression-based Bayesian Optimization
GPU	Graphics Processing Unit
GRU	Gated Recurrent Unit
HDAD	Histograms of Distance, Angle, Dihedral
HTML	HyperText Markup Language
ICSD	Inorganic Crystal Structure Database
IID	Independently and Identically Distributed
IR	InfraRed
Isomap	Isometric mapping
JSON	JavaScript Object Notation
Kadi4Mat	Karlsruhe Data Infrastructure for Materials Science
LASSO	Least Absolute Shrinkage and Selection Operator
LATP	Lithium Aluminum Titanium Phosphate
LDA	Linear Discriminant Analysis
LE	Laplacian Eigenmaps
LIBs	Lithium-Ion Batteries
LLE	Locally Linear Embedding
LLM	Large Language Model
LMAs	Lithium Metal Anodes
LR	Logistic Regression
LSTM	Long Short-Term Memory
MAGPIE	Materials Agnostic Platform for Informatics and Exploration
MAP	Maximum A Posteriori
MARAD	Molecular Atomic Radial Angular Distribution

MD	Molecular Dynamics
MEGNet	MatErials Graph Network
ML	Machine Learning
ML-IAP	Machine Learning Inter-Atomic Potential
MLE	Maximum Likelihood Estimation
MLP	Multi-Layer Perceptron
MP	Materials Project
MPNN	Message Passing Neural Network
MSE	Mean Squared Error
MVA	MultiVariate Analysis
NB	Naïve Bayes
NLP	Natural Language Processing
NMC	Nickel Manganese Cobalt
NMF	Non-negative Matrix Factorization
NOMAD	NOvel MAterials Discovery
NP	Nondeterministic Polynomial
NREL	National Renewable Energy Laboratory
OPTICS	Ordering Points To Identify Clustering Structure
OQMD	Open Quantum Materials Database
ORGANIC	Objective-Reinforced Generative Adversarial Network for Inverse-design Chemistry
OvR	One-versus-Rest
PCA	Principal Component Analysis
PDE	Partial Differential Equation
PDF	Probability Density Function or Pair Distribution Function

PFM	Phase-Field Method
PI	Probability of Improvement
PINN	Physics-Informed Neural Network
PR	Pattern Recognition
PR	Precision Recall
QM	Quantum Mechanical
RBF	Radial Basis Function
ReLU	Rectified Linear Unit
ResNet	Residual Network
RF	Random Forest
RL	Reinforcement Learning
RMSE	Root Mean Squared Error
RNN	Recurrent Neural Network
ROC	Receiver Operating Characteristic
SE	Squared Exponential
SEM	Scanning Electron Microscope
SGD	Stochastic Gradient Descent
SISSO	Sure Independence Screening and Sparsifying Operator
SMILES	Simplified Molecular Input Line-Entry System
SNE	Stochastic Neighbor Embedding
SOAP	Smooth Overlap of Atomic Positions
SOM	Self-Organizing Map
SP	Specific Peaks
SPM	Scanning Probe Microscopy
SRM	Structure Risk Minimization

STEM	Scanning Transmission Electron Microscopy
STING	STatistical INformation Grid
SVC	Support Vector machine Classifier
SVD	Singular Value Decomposition
SVM	Support Vector Machine
TL	Transfer Learning
TN	True Negative
ToF-SIMS	Time-of-Flight Secondary Ion Mass Spectrometry
TP	True Positive
UCB	Upper Confidence Bound
UMAP	Uniform Manifold Approximation and Projection
UV	UltraViolet
VAE	Variational AutoEncoder
XANES	X-ray Absorption Near-Edge Structure
XAS	X-ray Absorption Spectroscopy
XCT	X-ray Computer Tomography
XRD	X-Ray Diffraction

List of Figures

List of Tables

Bibliography

[1] Y. Liu, T. Zhao, W. Ju, and S. Shi. Materials discovery and design using machine learning. *Journal of Materiomics*, 3(3):159–177, 2017.

[2] K. Rajan. Materials informatics. *Materials Today*, 8(10):38–45, 2005.

[3] P. Hohenberg and W. Kohn. Inhomogeneous electron gas. *Physical Review*, 136(3B):B864, 1964.

[4] D. S. Sholl and J. A. Steckel. *Density functional theory: a practical introduction.* John Wiley & Sons, 2022.

[5] B. J. Alder and T. E. Wainwright. Studies in molecular dynamics. i. general method. *The Journal of Chemical Physics*, 31(2):459–466, 1959.

[6] A. Baumgärtner, A. Burkitt, D. Ceperley, H. De Raedt, A. Ferrenberg, D. Heermann, H. Herrmann, D. Landau, D. Levesque, W. von der Linden, et al. *The Monte Carlo method in condensed matter physics*, volume 71. Springer Science & Business Media, 2012.

[7] J. Hammersley. *Monte carlo methods.* Springer Science & Business Media, 2013.

[8] L.-Q. Chen. Phase-field models for microstructure evolution. *Annual Review of Materials Research*, 32(1):113–140, 2002.

[9] I. Steinbach. Phase-field models in materials science. *Modelling and Simulation in Materials Science and Engineering*, 17(7):073001, 2009.

[10] B. Nestler, H. Garcke, and B. Stinner. Multicomponent alloy solidification: phase-field modeling and simulations. *Physical Review E*, 71(4):041609, 2005.

[11] W. M. Lai, D. Rubin, and E. Krempl. *Introduction to continuum mechanics.* Butterworth-Heinemann, 2009.

[12] G. B. Olson. Designing a new material world. *Science*, 288(5468):993–998, 2000.

[13] S. Sagiroglu and D. Sinanc. Big data: a review. In *2013 International Conference on Collaboration Technologies and Systems (CTS)*, pages 42–47. IEEE, 2013.

[14] A. Belsky, M. Hellenbrandt, V. L. Karen, and P. Luksch. New developments in the inorganic crystal structure database (icsd): accessibility in support of materials research and design. *Acta Crystallographica Section B: Structural Science*, 58(3):364–369, 2002.

[15] A. Jain, S. P. Ong, G. Hautier, W. Chen, W. D. Richards, S. Dacek, S. Cholia, D. Gunter, D. Skinner, G. Ceder, et al. Commentary: the materials project: a materials genome approach to accelerating materials innovation. *APL Materials*, 1(1):011002, 2013.

[16] J. J. de Pablo, N. E. Jackson, M. A. Webb, L.-Q. Chen, J. E. Moore, D. Morgan, R. Jacobs, T. Pollock, D. G. Schlom, E. S. Toberer, et al. New frontiers for the materials genome initiative. *npj Computational Materials*, 5(1):41, 2019.

[17] C. Chen, Y. Zuo, W. Ye, X. Li, Z. Deng, and S. P. Ong. A critical review of machine learning of energy materials. *Advanced Energy Materials*, 10(8):1903242, 2020.

[18] K. M. Jablonka, D. Ongari, S. M. Moosavi, and B. Smit. Big-data science in porous materials: materials genomics and machine learning. *Chemical Reviews*, 120(16):8066–8129, 2020.

[19] W. Sha, Y. Li, S. Tang, J. Tian, Y. Zhao, Y. Guo, W. Zhang, X. Zhang, S. Lu, Y.-C. Cao, et al. Machine learning in polymer informatics. *InfoMat*, 3(4):353–361, 2021.

[20] A. G. Kusne, T. Gao, A. Mehta, L. Ke, M. C. Nguyen, K.-M. Ho, V. Antropov, C.-Z. Wang, M. J. Kramer, C. Long, et al. On-the-fly machine-learning for high-throughput experiments: search for rare-earth-free permanent magnets. *Scientific Reports*, 4(1):6367, 2014.

[21] T. M. Mitchell. *Machine learning.* McGraw-Hill, 1997.

[22] I. Goodfellow, Y. Bengio, and A. Courville. *Deep learning.* MIT press, 2016.

[23] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, et al. Deep neural networks for acoustic modeling in speech recognition: the shared views of four research groups. *IEEE Signal Processing Magazine*, 29(6):82–97, 2012.

[24] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems*, 25, 2012.

[25] A. Agrawal and A. Choudhary. Perspective: materials informatics and big data: realization of the "fourth paradigm" of science in materials science. *APL Materials*, 4(5), 2016.

[26] R. Jose and S. Ramakrishna. Materials 4.0: materials big data enabled materials discovery. *Applied Materials Today*, 10:127–132, 2018.

[27] A. O. Oliynyk and J. M. Buriak. Virtual issue on machine-learning discoveries in materials science. *Chemistry of Materials*, 31(20):8243–8247, 2019.

[28] W. Sha, Y. Guo, Q. Yuan, S. Tang, X. Zhang, S. Lu, X. Guo, Y.-C. Cao, and S. Cheng. Artificial intelligence to power the future of materials science and engineering. *Advanced Intelligent Systems*, 2(4):1900143, 2020.

[29] Y. Liu, Z. Yang, Z. Yu, Z. Liu, D. Liu, H. Lin, M. Li, S. Ma, M. Avdeev, and S. Shi. Generative artificial intelligence and its applications in materials science: current situation and future perspectives. *Journal of Materiomics*, 2023.

[30] 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):83, 2019.

[31] S. P. Ong. Accelerating materials science with high-throughput computations and machine learning. *Computational Materials Science*, 161:143–150, 2019.

[32] D. Morgan and R. Jacobs. Opportunities and challenges for machine learning in materials science. *arXiv preprint arXiv:2006.14604*, 2020.

[33] T. Lookman, P. V. Balachandran, D. Xue, and R. Yuan. Active learning in materials science with emphasis on adaptive sampling using uncertainties for targeted design. *npj Computational Materials*, 5(1):21, 2019.

[34] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[35] C. M. Bishop and N. M. Nasrabadi. *Pattern recognition and machine learning*, volume 4. Springer, 2006.

[36] T. Hastie, R. Tibshirani, J. H. Friedman, and J. H. Friedman. *The elements of statistical learning: data mining, inference, and prediction*, volume 2. Springer, 2009.

[37] S. Kullback and R. A. Leibler. On information and sufficiency. *The Annals of Mathematical Statistics*, 22(1):79–86, 1951.

[38] I. J. Good. Rational decisions. *Journal of the Royal Statistical Society: Series B (Methodological)*, 14(1):107–114, 1952.

[39] A. Nemirovski, A. Juditsky, G. Lan, and A. Shapiro. Robust stochastic approximation approach to stochastic programming. *SIAM Journal on Optimization*, 19(4):1574–1609, 2009.

[40] L. Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer, 2010.

[41] R. Bellman. Dynamic programming. *Science*, 153(3731):34–37, 1966.

[42] M. A. Hall. *Correlation-based feature selection for machine learning*. PhD thesis, The University of Waikato, 1999.

[43] S. Mika, G. Ratsch, J. Weston, B. Scholkopf, and K.-R. Mullers. Fisher discriminant analysis with kernels. In *Neural Networks for Signal Processing IX: Proceedings of the Signal Processing Society Workshop*, pages 41–48. IEEE, 1999.

[44] 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(11):559–572, 1901.

[45] I. T. Jolliffe and J. Cadima. Principal component analysis: a review and recent developments. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 374(2065):20150202, 2016.

[46] G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.

[47] 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(3):735–743, 2015.

[48] V. Stanev, C. Oses, A. G. Kusne, E. Rodriguez, J. Paglione, S. Curtarolo, and I. Takeuchi. Machine learning modeling of superconducting critical temperature. *npj Computational Materials*, 4(1):1–14, 2018.

[49] D. W. Hosmer Jr, S. Lemeshow, and R. X. Sturdivant. *Applied logistic regression*, volume 398. John Wiley & Sons, 2013.

[50] T. Cover and P. Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1):21–27, 1967.

[51] K. Q. Weinberger and L. K. Saul. Distance metric learning for large margin nearest neighbor classification. *Journal of Machine Learning Research*, 10(2), 2009.

[52] C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

[53] V. Vapnik. *The nature of statistical learning theory.* Springer Science & Business Media, 1999.

[54] J. Weston and C. Watkins. Support vector machines for multi-class pattern recognition. In *Esann*, volume 99, pages 219–224, 1999.

[55] K. Crammer and Y. Singer. On the algorithmic implementation of multiclass kernel-based vector machines. *Journal of Machine Learning Research*, 2:265–292, 2001.

[56] B. E. Boser, I. M. Guyon, and V. N. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the 5th Annual Workshop on Computational Learning Theory*, pages 144–152, 1992.

[57] C. K. Williams and C. E. Rasmussen. *Gaussian processes for machine learning*, volume 2. MIT press, 2006.

[58] J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1(1):81–106, 1986.

[59] J. R. Quinlan. *C4.5: programs for machine learning.* Morgan Kaufmann Publishers Inc., 1993.

[60] W.-Y. Loh. Classification and regression trees. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 1(1):14–23, 2011.

[61] B. Liu. *Web data mining: exploring hyperlinks, contents, and usage data*, volume 1. Springer, 2011.

[62] Z.-H. Zhou. *Ensemble methods: foundations and algorithms.* CRC press, 2012.

[63] M. Kearns and L. Valiant. Cryptographic limitations on learning boolean formulae and finite automata. *Journal of the ACM (JACM)*, 41(1):67–95, 1994.

[64] Y. Freund, R. Schapire, and N. Abe. A short introduction to boosting. *Journal-Japanese Society For Artificial Intelligence*, 14(771-780):1612, 1999.

[65] L. Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

[66] D. H. Wolpert. Stacked generalization. *Neural Networks*, 5(2):241–259, 1992.

[67] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.

[68] J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors). *The Annals of Statistics*, 28(2):337–407, 2000.

[69] J. H. Friedman. Greedy function approximation: a gradient boosting machine. *Annals of Statistics*, pages 1189–1232, 2001.

[70] T. Chen and C. Guestrin. Xgboost: a scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, page 785–794, 2016.

[71] B. Efron. Bootstrap methods: another look at the jackknife. In *Breakthroughs in Statistics: Methodology and Distribution*, pages 569–593. Springer, 1992.

[72] T. K. Ho. Random decision forests. In *Proceedings of 3rd International Conference on Document Analysis and Recognition*, volume 1, pages 278–282. IEEE, 1995.

[73] L. Breiman. Random forests. *Machine Learning*, 45:5–32, 2001.

[74] J. Han, J. Pei, and H. Tong. *Data mining: concepts and techniques.* Morgan Kaufmann Publishers Inc., 2022.

[75] A. K. Jain and R. C. Dubes. *Algorithms for clustering data.* Prentice-Hall Inc., 1988.

[76] J. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297, 1967.

[77] G. Gan, C. Ma, and J. Wu. *Data clustering: theory, algorithms, and applications.* SIAM, 2020.

[78] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, volume 96, pages 226–231, 1996.

[79] M. Ankerst, M. M. Breunig, H.-P. Kriegel, and J. Sander. Optics: ordering points to identify the clustering structure. *ACM Sigmod Record*, 28(2):49–60, 1999.

[80] W. Wang, J. Yang, and R. Muntz. Sting: a statistical information grid approach to spatial data mining. In *Proceedings of the 23rd International Conference on Very Large Data Bases*, volume 97, pages 186–195, 1997.

[81] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan. Automatic subspace clustering of high dimensional data for data mining applications. In *Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data*, pages 94–105, 1998.

[82] D. A. Reynolds. Gaussian mixture models. *Encyclopedia of Biometrics*, 741(659-663), 2009.

[83] T. Kohonen. The self-organizing map. *Proceedings of the IEEE*, 78(9):1464–1480, 1990.

[84] S. P. Boyd and L. Vandenberghe. *Convex optimization.* Cambridge University Press, 2004.

[85] J. B. Tenenbaum, V. d. Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, 2000.

[86] S. T. Roweis and L. K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326, 2000.

[87] M. Belkin and P. Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, 15(6):1373–1396, 2003.

[88] G. E. Hinton and S. Roweis. Stochastic neighbor embedding. *Advances in Neural Information Processing Systems*, 15, 2002.

[89] L. Van der Maaten and G. Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(11), 2008.

[90] L. McInnes, J. Healy, and J. Melville. Umap: uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*, 2018.

[91] P. Paatero and U. Tapper. Positive matrix factorization: a non-negative factor model with optimal utilization of error estimates of data values. *Environmetrics*, 5(2):111–126, 1994.

[92] D. D. Lee and H. S. Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791, 1999.

[93] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. De Freitas. Taking the human out of the loop: a review of bayesian optimization. *Proceedings of the IEEE*, 104(1):148–175, 2015.

[94] J. Snoek, H. Larochelle, and R. P. Adams. Practical bayesian optimization of machine learning algorithms. *arXiv preprint arXiv:1206.2944*, 2012.

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

[96] P. I. Frazier. A tutorial on bayesian optimization. *arXiv preprint arXiv:1807.02811*, 2018.

[97] H. J. Kushner. A new method of locating the maximum point of an arbitrary multipeak curve in the presence of noise. *Journal of Basic Engineering*, 1964.

[98] J. Močkus. On bayesian methods for seeking the extremum. In *Optimization Techniques IFIP Technical Conference*, pages 400–404. Springer, 1975.

[99] D. R. Jones, M. Schonlau, and W. J. Welch. Efficient global optimization of expensive black-box functions. *Journal of Global optimization*, 13(4):455–492, 1998.

[100] P. Auer. Using confidence bounds for exploitation-exploration trade-offs. *Journal of Machine Learning Research*, 3:397–422, 2002.

[101] Y. Gal and Z. Ghahramani. Dropout as a bayesian approximation: representing model uncertainty in deep learning. In *International Conference on Machine Learning*, pages 1050–1059. PMLR, 2016.

[102] R. Bellman. A markovian decision process. *Journal of Mathematics and Mechanics*, pages 679–684, 1957.

[103] R. S. Sutton and A. G. Barto. *Reinforcement learning: an introduction.* MIT press, 2018.

[104] M. Minsky. Steps toward artificial intelligence. *Proceedings of the IRE*, 49(1):8–30, 1961.

[105] F. Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386, 1958.

[106] P. Werbos. *Beyond regression: new tools for prediction and analysis in the behavioral sciences.* PhD thesis, Harvard University, 1974.

[107] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986.

[108] G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 2(4):303–314, 1989.

[109] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989.

[110] J. Zhou, G. Cui, S. Hu, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, and M. Sun. Graph neural networks: a review of methods and applications. *AI Open*, 1:57–81, 2020.

[111] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[112] 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*, pages 1263–1272. PMLR, 2017.

[113] V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 807–814, 2010.

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

[115] A. Maas, A. Hannun, and A. Ng. Rectifier nonlinearities improve neural network acoustic models. In *Proceedings of the 30th International Conference on Machine Learning*, volume 30, page 3, 2013.

[116] D.-A. Clevert, T. Unterthiner, and S. Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*, 2015.

[117] P. Ramachandran, B. Zoph, and Q. V. Le. Searching for activation functions. *arXiv preprint arXiv:1710.05941*, 2017.

[118] D. Hendrycks and K. Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.

[119] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[120] Y. LeCun, K. Kavukcuoglu, and C. Farabet. Convolutional networks and applications in vision. In *Proceedings of 2010 IEEE International Symposium on Circuits and Systems*, pages 253–256. IEEE, 2010.

[121] D. H. Hubel and T. N. Wiesel. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of Physiology*, 160(1):106, 1962.

[122] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[123] 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*, pages 770–778, 2016.

[124] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[125] O. Ronneberger, P. Fischer, and T. Brox. U-net: convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015: 18th International Conference*, pages 234–241. Springer, 2015.

[126] N. Siddique, S. Paheding, C. P. Elkin, and V. Devabhaktuni. U-net and its variants for medical image segmentation: a review of theory and applications. *IEEE Access*, 9:82031–82057, 2021.

[127] Y. Yu, X. Si, C. Hu, and J. Zhang. A review of recurrent neural networks: lstm cells and network architectures. *Neural Computation*, 31(7):1235–1270, 2019.

[128] P. J. Werbos. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560, 1990.

[129] Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166, 1994.

[130] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[131] K. Cho, B. Van Merriënboer, D. Bahdanau, and Y. Bengio. On the properties of neural machine translation: encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*, 2014.

[132] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[133] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[134] A. Zhang, Z. C. Lipton, M. Li, and A. J. Smola. *Dive into deep learning*. Cambridge University Press, 2023.

[135] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. *Advances in Neural Information Processing Systems*, 30, 2017.

[136] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 32(1):4–24, 2020.

[137] Z. Zhang, P. Cui, and W. Zhu. Deep learning on graphs: a survey. *IEEE Transactions on Knowledge and Data Engineering*, 34(1):249–270, 2020.

[138] D. P. Kingma and M. Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[139] D. J. Rezende, S. Mohamed, and D. Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *International Conference on Machine Learning*, pages 1278–1286. PMLR, 2014.

[140] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020.

[141] D. Rezende and S. Mohamed. Variational inference with normalizing flows. In *International Conference on Machine Learning*, pages 1530–1538. PMLR, 2015.

[142] J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, and S. Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International Conference on Machine Learning*, pages 2256–2265. PMLR, 2015.

[143] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever. Improving language understanding by generative pre-training. `https://openai.com/research/language-unsupervised`, 2018. Accessed: 2023-12-01.

[144] I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, and A. Lerchner. beta-vae: learning basic visual concepts with a constrained variational framework. In *International Conference on Learning Representations*, 2016.

[145] C. P. Burgess, I. Higgins, A. Pal, L. Matthey, N. Watters, G. Desjardins, and A. Lerchner. Understanding disentangling in β-vae. *arXiv preprint arXiv:1804.03599*, 2018.

[146] K. Sohn, H. Lee, and X. Yan. Learning structured output representation using deep conditional generative models. *Advances in Neural Information Processing Systems*, 28, 2015.

[147] A. Van Den Oord and O. Vinyals. Neural discrete representation learning. *Advances in Neural Information Processing Systems*, 30, 2017.

[148] A. B. L. Larsen, S. K. Sønderby, H. Larochelle, and O. Winther. Autoencoding beyond pixels using a learned similarity metric. In *International Conference on Machine Learning*, pages 1558–1566. PMLR, 2016.

[149] I. Goodfellow. Nips 2016 tutorial: generative adversarial networks. *arXiv preprint arXiv:1701.00160*, 2016.

[150] M. J. Osborne and A. Rubinstein. *A course in game theory*. MIT press, 1994.

[151] L. Mescheder, A. Geiger, and S. Nowozin. Which training methods for gans do actually converge? In *International Conference on Machine Learning*, pages 3481–3490. PMLR, 2018.

[152] M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein generative adversarial networks. In *International Conference on Machine Learning*, pages 214–223. PMLR, 2017.

[153] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[154] M. Mirza and S. Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

[155] A. Odena, C. Olah, and J. Shlens. Conditional image synthesis with auxiliary classifier gans. In *International Conference on Machine Learning*, pages 2642–2651. PMLR, 2017.

[156] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel. Infogan: interpretable representation learning by information maximizing generative adversarial nets. *Advances in Neural Information Processing Systems*, 29, 2016.

[157] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1125–1134, 2017.

[158] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2223–2232, 2017.

[159] S. J. Russell and P. Norvig. *Artificial intelligence a modern approach.* Pearson Education Inc., 2010.

[160] A. Zakutayev, N. Wunder, M. Schwarting, J. D. Perkins, R. White, K. Munch, W. Tumas, and C. Phillips. An open experimental database for exploring inorganic materials. *Scientific Data*, 5(1):1–12, 2018.

[161] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al. Scikit-learn: machine learning in python. *the Journal of Machine Learning Research*, 12:2825–2830, 2011.

[162] P. Villars, N. Onodera, and S. Iwata. The linus pauling file (lpf) and its application to materials design. *Journal of Alloys and Compounds*, 279(1):1–7, 1998.

[163] G. Bergerhoff, R. Hundt, R. Sievers, and I. Brown. The inorganic crystal structure data base. *Journal of Chemical Information and Computer Sciences*, 23(2):66–69, 1983.

[164] P. Villars and K. Cenzual. *Pearson's crystal data®: crystal structure database for inorganic compounds.* ASM International Materials Park, 2007.

[165] C. R. Groom, I. J. Bruno, M. P. Lightfoot, and S. C. Ward. The cambridge structural database. *Acta Crystallographica Section B: Structural Science, Crystal Engineering and Materials*, 72(2):171–179, 2016.

[166] S. Gražulis, D. Chateigner, R. T. Downs, A. Yokochi, M. Quirós, L. Lutterotti, E. Manakova, J. Butkus, P. Moeck, and A. Le Bail. Crystallography open database–an open-access collection of crystal structures. *Journal of Applied Crystallography*, 42(4):726–729, 2009.

[167] H. M. Berman, J. Westbrook, Z. Feng, G. Gilliland, T. N. Bhat, H. Weissig, I. N. Shindyalov, and P. E. Bourne. The protein data bank. *Nucleic Acids Research*, 28(1):235–242, 2000.

[168] J. J. Irwin and B. K. Shoichet. Zinc- a free database of commercially available compounds for virtual screening. *Journal of Chemical Information and Modeling*, 45(1):177–182, 2005.

[169] L. Ruddigkeit, R. Van Deursen, L. C. Blum, and J.-L. Reymond. Enumeration of 166 billion organic small molecules in the chemical universe database gdb-17. *Journal of Chemical Information and Modeling*, 52(11):2864–2875, 2012.

[170] S. Kim, J. Chen, T. Cheng, A. Gindulyte, J. He, S. He, Q. Li, B. A. Shoemaker, P. A. Thiessen, B. Yu, et al. Pubchem 2019 update: improved access to chemical data. *Nucleic Acids Research*, 47(D1):D1102–D1109, 2019.

[171] D. R. Lide. *CRC handbook of chemistry and physics*, volume 85. CRC press, 2004.

[172] W. Martienssen and H. Warlimont. *Springer handbook of condensed matter and materials data*, volume 1. Springer, 2005.

[173] A. N. Henderson, S. K. Kauwe, and T. D. Sparks. Benchmark datasets incorporating diverse tasks, sample sizes, material systems, and data heterogeneity for materials informatics. *Data in Brief*, 37:107262, 2021.

[174] K. Choudhary, B. DeCost, C. Chen, A. Jain, F. Tavazza, R. Cohn, C. W. Park, A. Choudhary, A. Agrawal, S. J. Billinge, et al. Recent advances and applications of deep learning methods in materials science. *npj Computational Materials*, 8(1):59, 2022.

[175] X. Jia, A. Lynch, Y. Huang, M. Danielson, I. Lang'at, A. Milder, A. E. Ruby, H. Wang, S. A. Friedler, A. J. Norquist, et al. Anthropogenic biases in chemical reaction data hinder exploratory inorganic synthesis. *Nature*, 573(7773):251–255, 2019.

[176] P. Raccuglia, K. C. Elbert, P. D. Adler, C. Falk, M. B. Wenny, A. Mollo, M. Zeller, S. A. Friedler, J. Schrier, and A. J. Norquist. Machine-learning-assisted materials discovery using failed experiments. *Nature*, 533(7601):73–76, 2016.

[177] M. Krallinger, O. Rabal, A. Lourenco, J. Oyarzabal, and A. Valencia. Information retrieval and text mining technologies for chemistry. *Chemical Reviews*, 117(12):7673–7761, 2017.

[178] M. C. Swain and J. M. Cole. Chemdataextractor: a toolkit for automated extraction of chemical information from the scientific literature. *Journal of Chemical Information and Modeling*, 56(10):1894–1904, 2016.

[179] E. Kim, K. Huang, A. Saunders, A. McCallum, G. Ceder, and E. Olivetti. Materials synthesis insights from scientific literature via text extraction and machine learning. *Chemistry of Materials*, 29(21):9436–9444, 2017.

[180] V. Tshitoyan, J. Dagdelen, L. Weston, A. Dunn, Z. Rong, O. Kononova, K. A. Persson, G. Ceder, and A. Jain. Unsupervised word embeddings capture latent knowledge from materials science literature. *Nature*, 571(7763):95–98, 2019.

[181] W. F. Maier, K. Stoewe, and S. Sieg. Combinatorial and high-throughput materials science. *Angewandte Chemie International Edition*, 46(32):6016–6067, 2007.

[182] S. Curtarolo, G. L. Hart, M. B. Nardelli, N. Mingo, S. Sanvito, and O. Levy. The high-throughput highway to computational materials design. *Nature Materials*, 12(3):191–201, 2013.

[183] S. Curtarolo, W. Setyawan, S. Wang, J. Xue, K. Yang, R. H. Taylor, L. J. Nelson, G. L. Hart, S. Sanvito, M. Buongiorno-Nardelli, et al. Aflowlib. org: a distributed materials properties repository from high-throughput ab initio calculations. *Computational Materials Science*, 58:227–235, 2012.

[184] D. D. Landis, J. S. Hummelshøj, S. Nestorov, J. Greeley, M. Dułak, T. Bligaard, J. K. Nørskov, and K. W. Jacobsen. The computational materials repository. *Computing in Science & Engineering*, 14(6):51–57, 2012.

[185] S. Kirklin, J. E. Saal, B. Meredig, A. Thompson, J. W. Doak, M. Aykol, S. Rühl, and C. Wolverton. The open quantum materials database (oqmd): assessing the accuracy of dft formation energies. *npj Computational Materials*, 1(1):1–15, 2015.

[186] C. Draxl and M. Scheffler. Nomad: the fair concept for big data-driven materials science. *MRS Bulletin*, 43(9):676–682, 2018.

[187] G. Pizzi, A. Cepellotti, R. Sabatini, N. Marzari, and B. Kozinsky. Aiida: automated interactive infrastructure and database for computational science. *Computational Materials Science*, 111:218–230, 2016.

[188] S. P. Ong, S. Cholia, A. Jain, M. Brafman, D. Gunter, G. Ceder, and K. A. Persson. The materials application programming interface (api): a simple, flexible and efficient api for materials data based on representational state transfer (rest) principles. *Computational Materials Science*, 97:209–215, 2015.

[189] R. H. Taylor, F. Rose, C. Toher, O. Levy, K. Yang, M. B. Nardelli, and S. Curtarolo. A restful api for exchanging materials data in the aflowlib. org consortium. *Computational Materials Science*, 93:178–192, 2014.

[190] J. Hachmann, R. Olivares-Amaya, S. Atahan-Evrenk, C. Amador-Bedolla, R. S. Sánchez-Carrera, A. Gold-Parker, L. Vogt, A. M. Brockway, and A. Aspuru-Guzik. The harvard clean energy project: large-scale computational screening and design of organic photovoltaics on the world community grid. *The Journal of Physical Chemistry Letters*, 2(17):2241–2251, 2011.

[191] S. Lany. Band-structure calculations for the 3 d transition metal oxides in g w. *Physical Review B*, 87(8):085112, 2013.

[192] D. P. Tabor, L. M. Roch, S. K. Saikin, C. Kreisbeck, D. Sheberla, J. H. Montoya, S. Dwaraknath, M. Aykol, C. Ortiz, H. Tribukait, et al. Accelerating the discovery of materials for clean energy in the era of smart automation. *Nature Reviews Materials*, 3(5):5–20, 2018.

[193] K. T. Butler, D. W. Davies, H. Cartwright, O. Isayev, and A. Walsh. Machine learning for molecular and materials science. *Nature*, 559(7715):547–555, 2018.

[194] G. R. Schleder, A. C. Padilha, C. M. Acosta, M. Costa, and A. Fazzio. From dft to machine learning: recent approaches to materials science–a review. *Journal of Physics: Materials*, 2(3):032001, 2019.

[195] N. Kireeva and V. S. Pervov. Materials space of solid-state electrolytes: unraveling chemical composition–structure–ionic conductivity relationships in garnet-type metal oxides using cheminformatics virtual screening approaches. *Physical Chemistry Chemical Physics*, 19(31):20904–20918, 2017.

[196] C. Chen, Z. M. Baiyee, and F. Ciucci. Unraveling the effect of La a-site substitution on oxygen ion diffusion and oxygen catalysis in perovskite $BaFeO_3$ by data-mining molecular dynamics and density functional theory. *Physical Chemistry Chemical Physics*, 17(37):24011–24019, 2015.

[197] C. Chen, Z. Lu, and F. Ciucci. Data mining of molecular dynamics data reveals li diffusion characteristics in garnet $Li_7La_3Zr_2O_{12}$. *Scientific Reports*, 7(1):1–8, 2017.

[198] E. V. Podryabinkin and A. V. Shapeev. Active learning of linearly parametrized interatomic potentials. *Computational Materials Science*, 140:171–180, 2017.

[199] D. Xue, P. V. Balachandran, J. Hogden, J. Theiler, D. Xue, and T. Lookman. Accelerated search for materials with targeted properties by adaptive design. *Nature Communications*, 7(1):1–9, 2016.

[200] S. Gow, M. Niranjan, S. Kanza, and J. G. Frey. A review of reinforcement learning in chemistry. *Digital Discovery*, 2022.

[201] D. H. Wolpert. The lack of a priori distinctions between learning algorithms. *Neural Computation*, 8(7):1341–1390, 1996.

[202] R. B. Wexler, J. M. P. Martirez, and A. M. Rappe. Chemical pressure-driven enhancement of the hydrogen evolving activity of ni2p from nonmetal surface doping interpreted via machine learning. *Journal of the American Chemical Society*, 140(13):4678–4683, 2018.

[203] I. Takigawa, K.-i. Shimizu, K. Tsuda, and S. Takakusagi. Machine-learning prediction of the d-band center for metals and bimetals. *RSC Advances*, 6(58):52587–52595, 2016.

[204] A. Furmanchuk, J. E. Saal, J. W. Doak, G. B. Olson, A. Choudhary, and A. Agrawal. Prediction of seebeck coefficient for compounds without restriction to fixed stoichiometry: a machine learning approach. *Journal of Computational Chemistry*, 39(4):191–202, 2018.

[205] H. Sahu, W. Rao, A. Troisi, and H. Ma. Toward predicting efficiency of organic solar cells via machine learning and improved descriptors. *Advanced Energy Materials*, 8(24):1801032, 2018.

[206] T. Xie and J. C. Grossman. Hierarchical visualization of materials space with graph convolutional neural networks. *The Journal of Chemical Physics*, 149(17):174111, 2018.

[207] K. Tran and Z. W. Ulissi. Active learning across intermetallics to guide discovery of electrocatalysts for CO_2 reduction and H_2 evolution. *Nature Catalysis*, 1(9):696–703, 2018.

[208] Y. Zhuo, A. Mansouri Tehrani, A. O. Oliynyk, A. C. Duke, and J. Brgoch. Identifying an efficient, thermally robust inorganic phosphor host via machine learning. *Nature Communications*, 9(1):1–10, 2018.

[209] B. Cao, L. A. Adutwum, A. O. Oliynyk, E. J. Luber, B. C. Olsen, A. Mar, and J. M. Buriak. How to optimize materials and devices via design of experiments and machine learning: demonstration using organic photovoltaics. *ACS Nano*, 12(8):7434–7444, 2018.

[210] S. Lu, Q. Zhou, Y. Ouyang, Y. Guo, Q. Li, and J. Wang. Accelerated discovery of stable lead-free hybrid organic-inorganic perovskites via machine learning. *Nature Communications*, 9(1):1–8, 2018.

[211] K. Choudhary, M. Bercx, J. Jiang, R. Pachter, D. Lamoen, and F. Tavazza. Accelerated discovery of efficient solar cell materials using quantum and machine-learning methods. *Chemistry of Materials*, 31(15):5900–5908, 2019.

[212] R. Jalem, M. Nakayama, and T. Kasuga. An efficient rule-based screening approach for discovering fast lithium ion conductors using density functional theory and artificial neural networks. *Journal of Materials Chemistry A*, 2(3):720–734, 2014.

[213] R. Jalem, M. Kimura, M. Nakayama, and T. Kasuga. Informatics-aided density functional theory study on the li ion transport of tavorite-type $LiMTO_4F$ (M^{3+}-T^{5+}, M^{2+}-T^{6+}). *Journal of Chemical Information and Modeling*, 55(6):1158–1168, 2015.

[214] A. D. Sendek, Q. Yang, E. D. Cubuk, K.-A. N. Duerloo, Y. Cui, and E. J. Reed. Holistic computational structure screening of more than 12000 candidates for solid lithium-ion conductor materials. *Energy & Environmental Science*, 10(1):306–320, 2017.

[215] F. A. Faber, L. Hutchison, B. Huang, J. Gilmer, S. S. Schoenholz, G. E. Dahl, O. Vinyals, S. Kearnes, P. F. Riley, and O. A. Von Lilienfeld. Prediction errors of molecular machine learning models lower than hybrid dft error. *Journal of Chemical Theory and Computation*, 13(11):5255–5264, 2017.

[216] R. Ramprasad, R. Batra, G. Pilania, A. Mannodi-Kanakkithodi, and C. Kim. Machine learning in materials informatics: recent applications and prospects. *npj Computational Materials*, 3(1):54, 2017.

[217] 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(1):1–7, 2016.

[218] W. Ye, C. Chen, Z. Wang, I.-H. Chu, and S. P. Ong. Deep neural networks for accurate predictions of crystal stability. *Nature Communications*, 9(1):3800, 2018.

[219] F. A. Faber, A. Lindmaa, O. A. Von Lilienfeld, and R. Armiento. Machine learning energies of 2 million elpasolite (ABC_2D_6) crystals. *Physical Review Letters*, 117(13):135502, 2016.

[220] B. Meredig, A. Agrawal, S. Kirklin, J. E. Saal, J. W. Doak, A. Thompson, K. Zhang, A. Choudhary, and C. Wolverton. Combinatorial screening for new materials in unconstrained composition space with machine learning. *Physical Review B*, 89(9):094104, 2014.

[221] 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(1):1–13, 2018.

[222] A. P. Bartók, R. Kondor, and G. Csányi. On representing chemical environments. *Physical Review B*, 87(18):184115, 2013.

[223] 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(5):058301, 2012.

[224] B. Huang and O. A. Von Lilienfeld. Communication: understanding molecular representations in machine learning: the role of uniqueness and target similarity. *The Journal of Chemical Physics*, 145(16):161102, 2016.

[225] D. Weininger. Smiles, a chemical language and information system. 1. introduction to methodology and encoding rules. *Journal of Chemical Information and Computer Sciences*, 28(1):31–36, 1988.

[226] D. Rogers and M. Hahn. Extended-connectivity fingerprints. *Journal of Chemical Information and Modeling*, 50(5):742–754, 2010.

[227] K. Hansen, F. Biegler, R. Ramakrishnan, W. Pronobis, O. A. Von Lilienfeld, K.-R. Muller, and A. Tkatchenko. Machine learning predictions of molecular properties: accurate many-body potentials and nonlocality in chemical space. *The Journal of Physical Chemistry Letters*, 6(12):2326–2331, 2015.

[228] J. Behler and M. Parrinello. Generalized neural-network representation of high-dimensional potential-energy surfaces. *Physical Review Letters*, 98(14):146401, 2007.

[229] 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(13):136403, 2010.

[230] A. V. Shapeev. Moment tensor potentials: a class of systematically improvable interatomic potentials. *Multiscale Modeling & Simulation*, 14(3):1153–1173, 2016.

[231] C. Chen, W. Ye, Y. Zuo, C. Zheng, and S. P. Ong. Graph networks as a universal machine learning framework for molecules and crystals. *Chemistry of Materials*, 31(9):3564–3572, 2019.

[232] T. Xie and J. C. Grossman. Crystal graph convolutional neural networks for an accurate and interpretable prediction of material properties. *Physical Review Letters*, 120(14):145301, 2018.

[233] 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(24):241722, 2018.

[234] D. Bonchev. *Chemical graph theory: introduction and fundamentals*, volume 1. CRC Press, 1991.

[235] G. Landrum. Rdkit: open-source cheminformatics. `https://www.rdkit.org`, 2006. Accessed: 2023-10-23.

[236] 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.

[237] L. Ward, A. Dunn, A. Faghaninia, N. E. Zimmermann, S. Bajaj, Q. Wang, J. Montoya, J. Chen, K. Bystrom, M. Dylla, et al. Matminer: an open source toolkit for materials data mining. *Computational Materials Science*, 152:60–69, 2018.

[238] R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288, 1996.

[239] C. Kim, G. Pilania, and R. Ramprasad. From organized high-throughput data to phenomenological theory using machine learning: the example of dielectric breakdown. *Chemistry of Materials*, 28(5):1304–1311, 2016.

[240] C. Kim, G. Pilania, and R. Ramprasad. Machine learning assisted predictions of intrinsic dielectric breakdown strength of abx3 perovskites. *The Journal of Physical Chemistry C*, 120(27):14575–14580, 2016.

[241] B. R. Goldsmith, M. Boley, J. Vreeken, M. Scheffler, and L. M. Ghiringhelli. Uncovering structure-property relationships of materials by subgroup discovery. *New Journal of Physics*, 19(1):013031, 2017.

[242] 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(8):083802, 2018.

[243] J. Im, S. Lee, T.-W. Ko, H. W. Kim, Y. Hyon, and H. Chang. Identifying pb-free perovskites for solar cells by machine learning. *npj Computational Materials*, 5(1):37, 2019.

[244] J. Yang and V. Honavar. Feature subset selection using a genetic algorithm. *IEEE Intelligent Systems and their Applications*, 13(2):44–49, 1998.

[245] 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(10):105503, 2015.

[246] L. M. Ghiringhelli, J. Vybiral, E. Ahmetcik, R. Ouyang, S. V. Levchenko, C. Draxl, and M. Scheffler. Learning physical descriptors for materials science by compressed sensing. *New Journal of Physics*, 19(2):023017, 2017.

[247] T. Lookman, F. J. Alexander, and K. Rajan. *Information science for materials discovery and design*, volume 225. Springer, 2016.

[248] N. Wagner and J. M. Rondinelli. Theory-guided machine learning in materials science. *Frontiers in Materials*, 3:28, 2016.

[249] S. Wold, K. Esbensen, and P. Geladi. Principal component analysis. *Chemometrics and Intelligent Laboratory Systems*, 2(1-3):37–52, 1987.

[250] J. A. Lee and M. Verleysen. *Nonlinear dimensionality reduction*, volume 1. Springer, 2007.

[251] H. Park, R. Mall, F. H. Alharbi, S. Sanvito, N. Tabet, H. Bensmail, and F. El-Mellouhi. Exploring new approaches towards the formability of mixed-ion perovskites by dft and machine learning. *Physical Chemistry Chemical Physics*, 21(3):1078–1088, 2019.

[252] M. Nunez. Exploring materials band structure space with unsupervised machine learning. *Computational Materials Science*, 158:117–123, 2019.

[253] M. W. Dorrity, L. M. Saunders, C. Queitsch, S. Fields, and C. Trapnell. Dimensionality reduction by umap to visualize physical and genetic interactions. *Nature Communications*, 11(1):1537, 2020.

[254] A. Jain, G. Hautier, S. P. Ong, C. J. Moore, C. C. Fischer, K. A. Persson, and G. Ceder. Formation enthalpies by mixing gga and gga+ u calculations. *Physical Review B*, 84(4):045115, 2011.

[255] A. M. Deml, R. O'Hayre, C. Wolverton, and V. Stevanović. Predicting density functional theory total energies and enthalpies of formation of metal-nonmetal compounds by linear regression. *Physical Review B*, 93(8):085142, 2016.

[256] J. Schmidt, J. Shi, P. Borlido, L. Chen, S. Botti, and M. A. Marques. Predicting the thermodynamic stability of solids combining density functional theory and machine learning. *Chemistry of Materials*, 29(12):5090–5103, 2017.

[257] 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(1):15679, 2017.

[258] X. Zheng, P. Zheng, and R.-Z. Zhang. Machine learning material properties from the periodic table using convolutional neural networks. *Chemical Science*, 9(44):8426–8432, 2018.

[259] A. O. Oliynyk, E. Antono, T. D. Sparks, L. Ghadbeigi, M. W. Gaultois, B. Meredig, and A. Mar. High-throughput machine-learning-driven synthesis of full-heusler compounds. *Chemistry of Materials*, 28(20):7324–7331, 2016.

[260] P. V. Balachandran, A. A. Emery, J. E. Gubernatis, T. Lookman, C. Wolverton, and A. Zunger. Predictions of new ab o 3 perovskite compounds by combining machine learning and density functional theory. *Physical Review Materials*, 2(4):043802, 2018.

[261] G. Pilania, P. V. Balachandran, J. E. Gubernatis, and T. Lookman. Classification of abo3 perovskite solids: a machine learning study. *Acta Crystallographica Section B: Structural Science, Crystal Engineering and Materials*, 71(5):507–513, 2015.

[262] W. Li, R. Jacobs, and D. Morgan. Predicting the thermodynamic stability of perovskite oxides using machine learning models. *Computational Materials Science*, 150:454–463, 2018.

[263] K. Kim, L. Ward, J. He, A. Krishna, A. Agrawal, and C. Wolverton. Machine-learning-accelerated high-throughput materials screening: discovery of novel quaternary heusler compounds. *Physical Review Materials*, 2(12):123801, 2018.

[264] G. Hautier, C. C. Fischer, A. Jain, T. Mueller, and G. Ceder. Finding nature's missing ternary oxide compounds using machine learning and density functional theory. *Chemistry of Materials*, 22(12):3762–3767, 2010.

[265] G. Hautier, C. Fischer, V. Ehrlacher, A. Jain, and G. Ceder. Data mined ionic substitutions for the discovery of new compounds. *Inorganic Chemistry*, 50(2):656–663, 2011.

[266] C. C. Fischer, K. J. Tibbetts, D. Morgan, and G. Ceder. Predicting crystal structure by merging data mining with quantum mechanics. *Nature Materials*, 5(8):641–646, 2006.

[267] T. Yamashita, N. Sato, H. Kino, T. Miyake, K. Tsuda, and T. Oguchi. Crystal structure prediction accelerated by bayesian optimization. *Physical Review Materials*, 2(1):013803, 2018.

[268] J. Graser, S. K. Kauwe, and T. D. Sparks. Machine learning and energy minimization approaches for crystal structure predictions: a review and new horizons. *Chemistry of Materials*, 30(11):3601–3612, 2018.

[269] A. O. Oliynyk, L. A. Adutwum, J. J. Harynuk, and A. Mar. Classifying crystal structures of binary compounds ab through cluster resolution feature selection and support vector machine analysis. *Chemistry of Materials*, 28(18):6672–6681, 2016.

[270] A. Ziletti, D. Kumar, M. Scheffler, and L. M. Ghiringhelli. Insightful classification of crystal structures using deep learning. *Nature Communications*, 9(1):2775, 2018.

[271] F. Legrain, J. Carrete, A. van Roekeghem, S. Curtarolo, and N. Mingo. How chemical composition alone can predict vibrational free energies and entropies of solids. *Chemistry of Materials*, 29(15):6220–6227, 2017.

[272] B. Medasani, A. Gamst, H. Ding, W. Chen, K. A. Persson, M. Asta, A. Canning, and M. Haranczyk. Predicting defect behavior in b2 intermetallics by merging ab initio modeling and machine learning. *npj Computational Materials*, 2(1):1, 2016.

[273] T. O. Owolabi, K. O. Akande, and S. O. Olatunji. Estimation of superconducting transition temperature tc for superconductors of the doped mgb 2 system from the crystal lattice parameters using support vector regression. *Journal of Superconductivity and Novel Magnetism*, 28:75–81, 2015.

[274] B. Meredig, E. Antono, C. Church, M. Hutchinson, J. Ling, S. Paradiso, B. Blaiszik, I. Foster, B. Gibbons, J. Hattrick-Simpers, et al. Can machine learning identify the next high-temperature superconductor? examining extrapolation performance for materials discovery. *Molecular Systems Design & Engineering*, 3(5):819–825, 2018.

[275] A. Seko, T. Maekawa, K. Tsuda, and I. Tanaka. Machine learning with systematic density-functional theory calculations: application to melting temperatures of single-and binary-component solids. *Physical Review B*, 89(5):054303, 2014.

[276] A. Agrawal, P. D. Deshpande, A. Cecen, G. P. Basavarsu, A. N. Choudhary, and S. R. Kalidindi. Exploration of data science techniques to predict fatigue strength of steel from composition and processing parameters. *Integrating Materials and Manufacturing Innovation*, 3:90–108, 2014.

[277] R. Saunders, C. Butler, J. Michopoulos, D. Lagoudas, A. Elwany, and A. Bagchi. Mechanical behavior predictions of additively manufactured microstructures using functional gaussian process surrogates. *npj Computational Materials*, 7(1):81, 2021.

[278] M. De Jong, W. Chen, R. Notestine, K. Persson, G. Ceder, A. Jain, M. Asta, and A. Gamst. A statistical learning framework for materials science: application to elastic moduli of k-nary inorganic polycrystalline compounds. *Scientific Reports*, 6(1):1–11, 2016.

[279] S. Aryal, R. Sakidja, M. W. Barsoum, and W.-Y. Ching. A genomic approach to the stability, elastic, and electronic properties of the max phases. *Physica Status Solidi (B)*, 251(8):1480–1497, 2014.

[280] J. D. Evans and F.-X. Coudert. Predicting the mechanical properties of zeolite frameworks by machine learning. *Chemistry of Materials*, 29(18):7833–7839, 2017.

[281] A. Furmanchuk, A. Agrawal, and A. Choudhary. Predictive analytics for crystalline materials: bulk modulus. *RSC Advances*, 6(97):95246–95251, 2016.

[282] A. Mansouri Tehrani, A. O. Oliynyk, M. Parry, Z. Rizvi, S. Couper, F. Lin, L. Miyagi, T. D. Sparks, and J. Brgoch. Machine learning directed search for ultraincompressible, superhard materials. *Journal of the American Chemical Society*, 140(31):9844–9853, 2018.

[283] A. Seko, A. Togo, H. Hayashi, K. Tsuda, L. Chaput, and I. Tanaka. Prediction of low-thermal-conductivity compounds with first-principles anharmonic lattice-dynamics calculations and bayesian optimization. *Physical Review Letters*, 115(20):205901, 2015.

[284] Z. Li, X. Ma, and H. Xin. Feature engineering of machine-learning chemisorption models for catalyst design. *Catalysis Today*, 280:232–238, 2017.

[285] W. T. Hong, R. E. Welsch, and Y. Shao-Horn. Descriptors of oxygen-evolution activity for oxides: a statistical evaluation. *The Journal of Physical Chemistry C*, 120(1):78–86, 2016.

[286] P. Dey, J. Bible, S. Datta, S. Broderick, J. Jasinski, M. Sunkara, M. Menon, and K. Rajan. Informatics-aided bandgap engineering for solar materials. *Computational Materials Science*, 83:185–195, 2014.

[287] J. Lee, A. Seko, K. Shitara, K. Nakayama, and I. Tanaka. Prediction model of band gap for inorganic compounds by combination of density functional theory calculations and machine learning techniques. *Physical Review B*, 93(11):115104, 2016.

[288] G. Pilania, A. Mannodi-Kanakkithodi, B. Uberuaga, R. Ramprasad, J. Gubernatis, and T. Lookman. Machine learning bandgaps of double perovskites. *Scientific Reports*, 6(1):19375, 2016.

[289] G. Pilania, J. E. Gubernatis, and T. Lookman. Multi-fidelity machine learning models for accurate bandgap predictions of solids. *Computational Materials Science*, 129:156–163, 2017.

[290] Y. Zhuo, A. Mansouri Tehrani, and J. Brgoch. Predicting the band gaps of inorganic solids by machine learning. *The Journal of Physical Chemistry Letters*, 9(7):1668–1673, 2018.

[291] A. C. Rajan, A. Mishra, S. Satsangi, R. Vaish, H. Mizuseki, K.-R. Lee, and A. K. Singh. Machine-learning-assisted accurate band gap predictions of functionalized mxene. *Chemistry of Materials*, 30(12):4031–4038, 2018.

[292] T. D. Huan, S. Boggs, G. Teyssedre, C. Laurent, M. Cakmak, S. Kumar, and R. Ramprasad. Advanced polymeric dielectrics for high energy density applications. *Progress in Materials Science*, 83:236–269, 2016.

[293] T. D. Huan, A. Mannodi-Kanakkithodi, C. Kim, V. Sharma, G. Pilania, and R. Ramprasad. A polymer dataset for accelerated property prediction and design. *Scientific Data*, 3(1):1–10, 2016.

[294] A. Mannodi-Kanakkithodi, G. M. Treich, T. D. Huan, R. Ma, M. Tefferi, Y. Cao, G. A. Sotzing, and R. Ramprasad. Rational co-design of polymer dielectrics for energy storage. *Advanced Materials*, 28(30):6277–6291, 2016.

[295] V. Sharma, C. Wang, R. G. Lorenzini, R. Ma, Q. Zhu, D. W. Sinkovits, G. Pilania, A. R. Oganov, S. Kumar, G. A. Sotzing, et al. Rational design of all organic polymer dielectrics. *Nature Communications*, 5(1):4845, 2014.

[296] B. Sanchez-Lengeling and A. Aspuru-Guzik. Inverse molecular design using machine learning: generative models for matter engineering. *Science*, 361(6400):360–365, 2018.

[297] A. Zunger. Inverse design in search of materials with target functionalities. *Nature Reviews Chemistry*, 2(4):0121, 2018.

[298] A. Nouira, N. Sokolovska, and J.-C. Crivello. Crystalgan: learning to discover crystallographic structures with generative adversarial networks. *arXiv preprint arXiv:1810.11203*, 2018.

[299] B. Sanchez-Lengeling, C. Outeiral, G. L. Guimaraes, and A. Aspuru-Guzik. Optimizing distributions over molecular space. an objective-reinforced generative adversarial network for inverse-design chemistry (organic). *ChemRxiv preprint chemrxiv:5309668.v3*, 2017.

[300] E. Putin, A. Asadulaev, Y. Ivanenkov, V. Aladinskiy, B. Sanchez-Lengeling, A. Aspuru-Guzik, and A. Zhavoronkov. Reinforced adversarial neural computer for de novo molecular design. *Journal of Chemical Information and Modeling*, 58(6):1194–1204, 2018.

[301] R. Gómez-Bombarelli, J. N. Wei, D. Duvenaud, J. M. Hernández-Lobato, B. Sánchez-Lengeling, D. Sheberla, J. Aguilera-Iparraguirre, T. D. Hirzel, R. P. Adams, and A. Aspuru-Guzik. Automatic chemical design using a data-driven continuous representation of molecules. *ACS Central Science*, 4(2):268–276, 2018.

[302] P. V. Balachandran, D. Xue, J. Theiler, J. Hogden, and T. Lookman. Adaptive strategies for materials design using uncertainties. *Scientific Reports*, 6(1):19660, 2016.

[303] A.-T. Nguyen, S. Reiter, and P. Rigo. A review on simulation-based optimization methods applied to building performance analysis. *Applied Energy*, 113:1043–1058, 2014.

[304] T. Lookman, P. V. Balachandran, D. Xue, J. Hogden, and J. Theiler. Statistical inference and adaptive design for materials discovery. *Current Opinion in Solid State and Materials Science*, 21(3):121–128, 2017.

[305] W. B. Powell and I. O. Ryzhov. *Optimal learning*, volume 841. John Wiley & Sons, 2012.

[306] A. Kendall and Y. Gal. What uncertainties do we need in bayesian deep learning for computer vision? *Advances in Neural Information Processing Systems*, 30, 2017.

[307] B. Efron and R. J. Tibshirani. *An introduction to the bootstrap.* CRC press, 1994.

[308] T. J. DiCiccio and B. Efron. Bootstrap confidence intervals. *Statistical Science*, 11(3):189–228, 1996.

[309] 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(1):1929–1958, 2014.

[310] J. Behler. Representing potential energy surfaces by high-dimensional neural network potentials. *Journal of Physics: Condensed Matter*, 26(18):183001, 2014.

[311] C. Kim, A. Chandrasekaran, A. Jha, and R. Ramprasad. Active-learning and materials design: the example of high glass transition temperature polymers. *MRS Communications*, 9(3):860–866, 2019.

[312] K. Homma, Y. Liu, M. Sumita, R. Tamura, N. Fushimi, J. Iwata, K. Tsuda, and C. Kaneta. Optimization of a heterogeneous ternary li3po4–li3bo3–li2so4 mixture for li-ion conductivity by machine learning. *The Journal of Physical Chemistry C*, 124(24):12865–12870, 2020.

[313] M. Harada, H. Takeda, S. Suzuki, K. Nakano, N. Tanibata, M. Nakayama, M. Karasuyama, and I. Takeuchi. Bayesian-optimization-guided experimental search of nasicon-type solid electrolytes for all-solid-state Li-ion batteries. *Journal of Materials Chemistry A*, 8(30):15103–15109, 2020.

[314] Z. Yang, S. Suzuki, N. Tanibata, H. Takeda, M. Nakayama, M. Karasuyama, and I. Takeuchi. Efficient experimental search for discovering a fast Li-ion conductor from a perovskite-type $\mathrm{Li_xLa_{(1\text{-}x)/3}Ti_{1.7}NbO_3}$ (llno) solid-state electrolyte using bayesian optimization. *The Journal of Physical Chemistry C*, 125(1):152–160, 2020.

[315] Q. Song, Y. Bai, and Q. Chen. The spring of processing chemistry in perovskite solar cells–bayesian optimization. *The Journal of Physical Chemistry Letters*, 13(46):10741–10750, 2022.

[316] R. Gómez-Bombarelli, J. Aguilera-Iparraguirre, T. D. Hirzel, D. Duvenaud, D. Maclaurin, M. A. Blood-Forsythe, H. S. Chae, M. Einzinger, D.-G. Ha, T. Wu, et al. Design of efficient molecular organic light-emitting diodes by a high-throughput virtual screening and experimental approach. *Nature Materials*, 15(10):1120–1127, 2016.

[317] B. J. Shields, J. Stevens, J. Li, M. Parasram, F. Damani, J. I. M. Alvarado, J. M. Janey, R. P. Adams, and A. G. Doyle. Bayesian reaction optimization as a tool for chemical synthesis. *Nature*, 590(7844):89–96, 2021.

[318] Z. Rao, P.-Y. Tung, R. Xie, Y. Wei, H. Zhang, A. Ferrari, T. Klaver, F. Körmann, P. T. Sukumar, A. Kwiatkowski da Silva, et al. Machine learning–enabled high-entropy alloy discovery. *Science*, 378(6615):78–85, 2022.

[319] J. Gubernatis and T. Lookman. Machine learning in materials design and discovery: examples from the present and suggestions for the future. *Physical Review Materials*, 2(12):120301, 2018.

[320] K. Terayama, M. Sumita, R. Tamura, and K. Tsuda. Black-box optimization for automated discovery. *Accounts of Chemical Research*, 54(6):1334–1346, 2021.

[321] 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(1):498–503, 2017.

[322] B. P. MacLeod, F. G. Parlane, T. D. Morrissey, F. Häse, L. M. Roch, K. E. Dettelbach, R. Moreira, L. P. Yunker, M. B. Rooney, J. R. Deeth, et al. Self-driving laboratory for accelerated discovery of thin-film materials. *Science Advances*, 6(20):eaaz8867, 2020.

[323] X. Du, L. Lüer, T. Heumueller, J. Wagner, C. Berger, T. Osterrieder, J. Wortmann, S. Langner, U. Vongsaysy, M. Bertrand, et al. Elucidating the full potential of opv materials utilizing a high-throughput robot-based platform and machine learning. *Joule*, 5(2):495–506, 2021.

[324] Y. Jiang, D. Salley, A. Sharma, G. Keenan, M. Mullin, and L. Cronin. An artificial intelligence enabled chemical synthesis robot for exploration and optimization of nanomaterials. *Science Advances*, 8(40):eabo2626, 2022.

[325] V. Duros, J. Grizou, W. Xuan, Z. Hosni, D.-L. Long, H. N. Miras, and L. Cronin. Human versus robots in the discovery and crystallization of gigantic polyoxometalates. *Angewandte Chemie*, 129(36):10955–10960, 2017.

[326] J. M. Granda, L. Donina, V. Dragone, D.-L. Long, and L. Cronin. Controlling an organic synthesis robot with machine learning to search for new reactivity. *Nature*, 559(7714):377–381, 2018.

[327] P. Nikolaev, D. Hooper, F. Webber, R. Rao, K. Decker, M. Krein, J. Poleski, R. Barto, and B. Maruyama. Autonomy in materials research: a case study in carbon nanotube growth. *npj Computational Materials*, 2(1):1–6, 2016.

[328] A. Koeppe. *Deep learning in the finite element method.* PhD thesis, Dissertation, Rheinisch-Westfälische Technische Hochschule Aachen, 2021, 2021.

[329] D. Montes de Oca Zapiain, J. A. Stewart, and R. Dingreville. Accelerating phase-field-based microstructure evolution predictions via surrogate models trained by machine learning methods. *npj Computational Materials*, 7(1):3, 2021.

[330] J. C. Snyder, M. Rupp, K. Hansen, K.-R. Müller, and K. Burke. Finding density functionals with machine learning. *Physical Review Letters*, 108(25):253002, 2012.

[331] J. C. Snyder, M. Rupp, K. Hansen, L. Blooston, K.-R. Müller, and K. Burke. Orbital-free bond breaking via machine learning. *The Journal of Chemical Physics*, 139(22), 2013.

[332] V. L. Deringer and G. Csányi. Machine learning based interatomic potential for amorphous carbon. *Physical Review B*, 95(9):094203, 2017.

[333] F. Brockherde, L. Vogt, L. Li, M. E. Tuckerman, K. Burke, and K.-R. Müller. Bypassing the kohn-sham equations with machine learning. *Nature Communications*, 8(1):872, 2017.

[334] J. Behler, R. Martoňák, D. Donadio, and M. Parrinello. Metadynamics simulations of the high-pressure phases of silicon employing a high-dimensional neural network potential. *Physical Review Letters*, 100(18):185501, 2008.

[335] S. Chmiela, A. Tkatchenko, H. E. Sauceda, I. Poltavsky, K. T. Schütt, and K.-R. Müller. Machine learning of accurate energy-conserving molecular force fields. *Science Advances*, 3(5):e1603015, 2017.

[336] W. J. Szlachta, A. P. Bartók, and G. Csányi. Accuracy and transferability of gaussian approximation potential models for tungsten. *Physical Review B*, 90(10):104108, 2014.

[337] A. P. Bartók and G. Csányi. G aussian approximation potentials: a brief tutorial introduction. *International Journal of Quantum Chemistry*, 115(16):1051–1057, 2015.

[338] S. Jindal, S. Chiriki, and S. S. Bulusu. Spherical harmonics based descriptor for neural network potentials: structure and dynamics of au147 nanocluster. *The Journal of Chemical Physics*, 146(20), 2017.

[339] A. P. Thompson, L. P. Swiler, C. R. Trott, S. M. Foiles, and G. J. Tucker. Spectral neighbor analysis method for automated generation of quantum-accurate interatomic potentials. *Journal of Computational Physics*, 285:316–330, 2015.

[340] T. Mueller, A. G. Kusne, and R. Ramprasad. Machine learning in materials science: recent progress and emerging applications. *Reviews in Computational Chemistry*, 29:186–273, 2016.

[341] M. Rupp. Machine learning for quantum mechanics in a nutshell. *International Journal of Quantum Chemistry*, 115(16):1058–1073, 2015.

[342] C. Hu, S. Martin, and R. Dingreville. Accelerating phase-field predictions via recurrent neural networks learning the microstructure evolution in latent space. *Computer Methods in Applied Mechanics and Engineering*, 397:115128, 2022.

[343] W. Yan, J. Melville, V. Yadav, K. Everett, L. Yang, M. S. Kesler, A. R. Krause, M. R. Tonks, and J. B. Harley. A novel physics-regularized interpretable machine learning model for grain growth. *Materials & Design*, 222:111032, 2022.

[344] Z. Wang, W. Yang, L. Xiang, X. Wang, Y. Zhao, Y. Xiao, P. Liu, Y. Liu, M. Banu, O. Zikanov, et al. Multi-input convolutional network for ultrafast simulation of field evolvement. *Patterns*, 3(6), 2022.

[345] K. Yang, Y. Cao, Y. Zhang, S. Fan, M. Tang, D. Aberg, B. Sadigh, and F. Zhou. Self-supervised learning and prediction of microstructure evolution with convolutional recurrent neural networks. *Patterns*, 2(5), 2021.

[346] I. Peivaste, N. H. Siboni, G. Alahyarizadeh, R. Ghaderi, B. Svendsen, D. Raabe, and J. R. Mianroodi. Accelerating phase-field-based simulation via machine learning. *arXiv preprint arXiv:2205.02121*, 2022.

[347] B.-Y. Tseng, C.-W. C. Guo, Y.-C. Chien, J.-P. Wang, and C.-H. Yu. Deep learning model to predict ice crystal growth. *Advanced Science*, page 2207731, 2023.

[348] Y. Qin, S. DeWitt, B. Radhakrishnan, and G. Biros. Grainnn: a neighbor-aware long short-term memory network for predicting microstructure evolution during polycrystalline grain formation. *Computational Materials Science*, 218:111927, 2023.

[349] A. A. K. Farizhandi and M. Mamivand. Spatiotemporal prediction of microstructure evolution with predictive recurrent neural network. *Computational Materials Science*, 223:112110, 2023.

[350] A. A. K. Farizhandi, O. Betancourt, and M. Mamivand. Deep learning approach for chemistry and processing history prediction from materials microstructure. *Scientific Reports*, 12(1):4552, 2022.

[351] J. R. Mianroodi, N. H. Siboni, and D. Raabe. Teaching solid mechanics to artificial intelligence—a fast solver for heterogeneous materials. *npj Computational Materials*, 7(1):99, 2021.

[352] A. I. Forrester, A. Sóbester, and A. J. Keane. Multi-fidelity optimization via surrogate modelling. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 463(2088):3251–3269, 2007.

[353] P. Perdikaris, D. Venturi, J. O. Royset, and G. E. Karniadakis. Multi-fidelity modelling via recursive co-kriging and gaussian–markov random fields. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 471(2179):20150018, 2015.

[354] M. Raissi, P. Perdikaris, and G. E. Karniadakis. Physics-informed neural networks: a deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019.

[355] G. E. Karniadakis, I. G. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, and L. Yang. Physics-informed machine learning. *Nature Reviews Physics*, 3(6):422–440, 2021.

[356] S. Cuomo, V. S. Di Cola, F. Giampaolo, G. Rozza, M. Raissi, and F. Piccialli. Scientific machine learning through physics–informed neural networks: where we are and what's next. *Journal of Scientific Computing*, 92(3):88, 2022.

[357] S. Cai, Z. Mao, Z. Wang, M. Yin, and G. E. Karniadakis. Physics-informed neural networks (pinns) for fluid mechanics: a review. *Acta Mechanica Sinica*, 37(12):1727–1738, 2021.

[358] L. Lu, X. Meng, Z. Mao, and G. E. Karniadakis. Deepxde: a deep learning library for solving differential equations. *SIAM Review*, 63(1):208–228, 2021.

[359] K. Shukla, P. C. Di Leoni, J. Blackshire, D. Sparkman, and G. E. Karniadakis. Physics-informed neural network for ultrasound nondestructive quantification of surface breaking cracks. *Journal of Nondestructive Evaluation*, 39:1–20, 2020.

[360] L. Lu, M. Dao, P. Kumar, U. Ramamurty, G. E. Karniadakis, and S. Suresh. Extraction of mechanical properties of materials through deep learning from instrumented indentation. *Proceedings of the National Academy of Sciences*, 117(13):7052–7062, 2020.

[361] E. Zhang, M. Dao, G. E. Karniadakis, and S. Suresh. Analyses of internal structures and defects in materials using physics-informed neural networks. *Science Advances*, 8(7):eabk0644, 2022.

[362] Z. Zhang and G. X. Gu. Physics-informed deep learning for digital materials. *Theoretical and Applied Mechanics Letters*, 11(1):100220, 2021.

[363] G. P. Pun, R. Batra, R. Ramprasad, and Y. Mishin. Physically informed artificial neural networks for atomistic modeling of materials. *Nature Communications*, 10(1):2339, 2019.

[364] Z. Hao, S. Liu, Y. Zhang, C. Ying, Y. Feng, H. Su, and J. Zhu. Physics-informed machine learning: a survey on problems, methods and applications. *arXiv preprint arXiv:2211.08064*, 2022.

[365] N. Thuerey, P. Holl, M. Mueller, P. Schnell, F. Trost, and K. Um. Physics-based deep learning. *arXiv preprint arXiv:2109.05237*, 2021.

[366] S. V. Kalinin, B. G. Sumpter, and R. K. Archibald. Big–deep–smart data in imaging for guiding materials design. *Nature Materials*, 14(10):973–980, 2015.

[367] W. B. Park, J. Chung, J. Jung, K. Sohn, S. P. Singh, M. Pyo, N. Shin, and K.-S. Sohn. Classification of crystal structure using a convolutional neural network. *The International Union of Crystallography Journal*, 4(4):486–494, 2017.

[368] J. Timoshenko and A. I. Frenkel. "inverting" x-ray absorption spectra of catalysts by machine learning in search for activity descriptors. *ACS Catalysis*, 9(11):10192–10211, 2019.

[369] A. N. Zaloga, V. V. Stanovov, O. E. Bezrukova, P. S. Dubinin, and I. S. Yakimov. Crystal symmetry classification from powder x-ray diffraction patterns using a convolutional neural network. *Materials Today Communications*, 25:101662, 2020.

[370] J. Schuetzke, N. J. Szymanski, and M. Reischl. Validating neural networks for spectroscopic classification on a universal synthetic dataset. *npj Computational Materials*, 9(1):100, 2023.

[371] J.-W. Lee, W. B. Park, J. H. Lee, S. P. Singh, and K.-S. Sohn. A deep-learning technique for phase identification in multiphase inorganic compounds using synthetic xrd powder patterns. *Nature Communications*, 11(1):86, 2020.

[372] H. Dong, K. T. Butler, D. Matras, S. W. Price, Y. Odarchenko, R. Khatry, A. Thompson, V. Middelkoop, S. D. Jacques, A. M. Beale, et al. A deep convolutional neural network for real-time full profile analysis of big powder diffraction data. *npj Computational Materials*, 7(1):74, 2021.

[373] M. Hellenbrandt. The inorganic crystal structure database (icsd)—present and future. *Crystallography Reviews*, 10(1):17–22, 2004.

[374] H. Wang, Y. Xie, D. Li, H. Deng, Y. Zhao, M. Xin, and J. Lin. Rapid identification of x-ray diffraction patterns based on very limited data by interpretable convolutional neural networks. *Journal of Chemical Information and Modeling*, 60(4):2004–2011, 2020.

[375] P. M. Maffettone, L. Banko, P. Cui, Y. Lysogorskiy, M. A. Little, D. Olds, A. Ludwig, and A. I. Cooper. Crystallography companion agent for high-throughput materials discovery. *Nature Computational Science*, 1(4):290–297, 2021.

[376] F. Oviedo, Z. Ren, S. Sun, C. Settens, Z. Liu, N. T. P. Hartono, S. Ramasamy, B. L. DeCost, S. I. Tian, G. Romano, et al. Fast and interpretable classification of small x-ray diffraction datasets using data augmentation and deep neural networks. *npj Computational Materials*, 5(1):60, 2019.

[377] J. A. Aguiar, M. L. Gong, and T. Tasdizen. Crystallographic prediction from diffraction and chemistry data for higher throughput classification using machine learning. *Computational Materials Science*, 173:109409, 2020.

[378] C. M. Fancher, Z. Han, I. Levin, K. Page, B. J. Reich, R. C. Smith, A. G. Wilson, and J. L. Jones. Use of bayesian inference in crystallographic structure refinement via full diffraction profile analysis. *Scientific Reports*, 6(1):31625, 2016.

[379] M. L. Green, C. Choi, J. Hattrick-Simpers, A. Joshi, I. Takeuchi, S. Barron, E. Campo, T. Chiang, S. Empedocles, J. Gregoire, et al. Fulfilling the promise of the materials genome initiative with high-throughput experimental methodologies. *Applied Physics Reviews*, 4(1), 2017.

[380] J. R. Hattrick-Simpers, J. M. Gregoire, and A. G. Kusne. Perspective: composition–structure–property mapping in high-throughput experiments: turning data into knowledge. *APL Materials*, 4(5), 2016.

[381] A. G. Kusne, D. Keller, A. Anderson, A. Zaban, and I. Takeuchi. High-throughput determination of structural phase diagram and constituent phases using grendel. *Nanotechnology*, 26(44):444002, 2015.

[382] J. K. Bunn, J. Hu, and J. R. Hattrick-Simpers. Semi-supervised approach to phase identification from combinatorial sample diffraction patterns. *JOM*, 68:2116–2125, 2016.

[383] N. J. Szymanski, C. J. Bartel, Y. Zeng, Q. Tu, and G. Ceder. Probabilistic deep learning approach to automate the interpretation of multi-phase diffraction spectra. *Chemistry of Materials*, 33(11):4204–4215, 2021.

[384] C. Zheng, K. Mathew, C. Chen, Y. Chen, H. Tang, A. Dozier, J. J. Kas, F. D. Vila, J. J. Rehr, L. F. Piper, et al. Automated generation and ensemble-learned matching of x-ray absorption spectra. *npj Computational Materials*, 4(1):12, 2018.

[385] J. Timoshenko, A. Anspoks, A. Cintins, A. Kuzmin, J. Purans, and A. I. Frenkel. Neural network approach for characterizing structural transformations by x-ray absorption fine structure spectroscopy. *Physical Review Letters*, 120(22):225502, 2018.

[386] C. Zheng, C. Chen, Y. Chen, and S. P. Ong. Random forest models for accurate identification of coordination environments from x-ray absorption near-edge structure. *Patterns*, 1(2), 2020.

[387] S. B. Torrisi, M. R. Carbone, B. A. Rohr, J. H. Montoya, Y. Ha, J. Yano, S. K. Suram, and L. Hung. Random forest machine learning models for interpretable x-ray absorption near-edge structure spectrum-property relationships. *npj Computational Materials*, 6(1):109, 2020.

[388] N. Andrejevic, J. Andrejevic, B. A. Bernevig, N. Regnault, F. Han, G. Fabbris, T. Nguyen, N. C. Drucker, C. H. Rycroft, and M. Li. Machine-learning spectral indicators of topology. *Advanced Materials*, 34(49):2204113, 2022.

[389] J. Yang, J. Xu, X. Zhang, C. Wu, T. Lin, and Y. Ying. Deep learning for vibrational spectral analysis: recent progress and a practical guide. *Analytica Chimica Acta*, 1081:6–17, 2019.

[390] M. G. Madden and A. G. Ryder. Machine learning methods for quantitative analysis of raman spectroscopy data. In *Opto-Ireland 2002: Optics and Photonics Technologies and Applications*, volume 4876, pages 1130–1139, 2003.

[391] J. Conroy, A. G. Ryder, M. N. Leger, K. Hennessey, and M. G. Madden. Qualitative and quantitative analysis of chlorinated solvents using raman spectroscopy and machine learning. In *Opto-Ireland 2005: Optical Sensing and Spectroscopy*, volume 5826, pages 131–142, 2005.

[392] J. Acquarelli, T. van Laarhoven, J. Gerretzen, T. N. Tran, L. M. Buydens, and E. Marchiori. Convolutional neural networks for vibrational spectroscopic data analysis. *Analytica Chimica Acta*, 954:22–31, 2017.

[393] M.-L. O'connell, T. Howley, A. G. Ryder, M. N. Leger, and M. G. Madden. Classification of a target analyte in solid mixtures using principal component analysis, support vector machines, and raman spectroscopy. In *Opto-Ireland 2005: Optical Sensing and Spectroscopy*, volume 5826, pages 340–350, 2005.

[394] J. Zhao, Q. Chen, X. Huang, and C. Fang. Qualitative identification of tea categories by near infrared spectroscopy and support vector machine. *Journal of Pharmaceutical and Biomedical Analysis*, 41(4):1198–1204, 2006.

[395] J. Liu, M. Osadchy, L. Ashton, M. Foster, C. J. Solomon, and S. J. Gibson. Deep convolutional neural networks for raman spectrum recognition: a unified solution. *Analyst*, 142(21):4067–4074, 2017.

[396] C.-H. Liu, Y. Tao, D. Hsu, Q. Du, and S. J. Billinge. Using a machine learning approach to determine the space group of a structure from the atomic pair distribution function. *Acta Crystallographica Section A: Foundations and Advances*, 75(4):633–643, 2019.

[397] P. Selzer, J. Gasteiger, H. Thomas, and R. Salzer. Rapid access to infrared reference spectra of arbitrary organic compounds: scope and limitations of an approach to the simulation of infrared spectra by neural networks. *Chemistry–A European Journal*, 6(5):920–927, 2000.

[398] K. Ghosh, A. Stuke, M. Todorović, P. B. Jørgensen, M. N. Schmidt, A. Vehtari, and P. Rinke. Deep learning spectroscopy: neural networks for molecular excitation spectra. *Advanced Science*, 6(9):1801367, 2019.

[399] T. Kostka, P. Selzer, and J. Gasteiger. A combined application of reaction prediction and infrared spectra simulation for the identification of degradation products of s-triazine herbicides. *Chemistry–A European Journal*, 7(10):2254–2260, 2001.

[400] C. B. Mahmoud, A. Anelli, G. Csányi, and M. Ceriotti. Learning the electronic density of states in condensed matter. *Physical Review B*, 102(23):235130, 2020.

[401] S. Kong, F. Ricci, D. Guevarra, J. B. Neaton, C. P. Gomes, and J. M. Gregoire. Density of states prediction for materials discovery via contrastive learning from probabilistic embeddings. *Nature Communications*, 13(1):949, 2022.

[402] Z. Chen, N. Andrejevic, T. Smidt, Z. Ding, Q. Xu, Y.-T. Chi, Q. T. Nguyen, A. Alatas, J. Kong, and M. Li. Direct prediction of phonon density of states with euclidean neural networks. *Advanced Science*, 8(12):2004214, 2021.

[403] P. R. Kaundinya, K. Choudhary, and S. R. Kalidindi. Prediction of the electron density of states for crystalline compounds with atomistic line graph neural networks (alignn). *JOM*, 74(4):1395–1405, 2022.

[404] M. R. Carbone, M. Topsakal, D. Lu, and S. Yoo. Machine-learning x-ray absorption spectra to quantitative accuracy. *Physical Review Letters*, 124(15):156401, 2020.

[405] C. D. Rankine, M. M. Madkhali, and T. J. Penfold. A deep neural network for the rapid prediction of x-ray absorption spectra. *The Journal of Physical Chemistry A*, 124(21):4263–4270, 2020.

[406] V. Fung, G. Hu, P. Ganesh, and B. G. Sumpter. Machine learned features from density of states for accurate adsorption energy prediction. *Nature Communications*, 12(1):88, 2021.

[407] H. S. Stein, E. Soedarmadji, P. F. Newhouse, D. Guevarra, and J. M. Gregoire. Synthesis, optical imaging, and absorption spectroscopy data for 179072 metal oxides. *Scientific Data*, 6(1):9, 2019.

[408] K. Choudhary, T. Yildirim, D. W. Siderius, A. G. Kusne, A. McDannald, and D. L. Ortiz-Montalvo. Graph neural network predictions of metal organic framework co2 adsorption properties. *Computational Materials Science*, 210:111388, 2022.

[409] R. Anderson, A. Biong, and D. A. Gómez-Gualdrón. Adsorption isotherm predictions for multiple molecules in mofs using the same deep learning model. *Journal of Chemical Theory and Computation*, 16(2):1271–1283, 2020.

[410] N. Martin and H. Maes. *Multivariate analysis*. Academic Press, 1979.

[411] C.-H. Liu, C. J. Wright, R. Gu, S. Bandi, A. Wustrow, P. K. Todd, D. O'Nolan, M. L. Beauvais, J. R. Neilson, P. J. Chupas, et al. Validation of non-negative matrix factorization for assessment of atomic pair-distribution function (pdf) data in a real-time streaming context. *arXiv preprint arXiv:2010.11807*, 2020.

[412] D. J. Graham and D. G. Castner. Multivariate analysis of tof-sims data from multicomponent systems: the why, when, and how. *Biointerphases*, 7(1):49, 2012.

[413] E. A. Holm, R. Cohn, N. Gao, A. R. Kitahara, T. P. Matson, B. Lei, and S. R. Yarasi. Overview: computer vision and machine learning for microstructural characterization and analysis. *Metallurgical and Materials Transactions A*, 51:5985–5999, 2020.

[414] M. H. Modarres, R. Aversa, S. Cozzini, R. Ciancio, A. Leto, and G. P. Brandino. Neural network for nanoscience scanning electron microscope image recognition. *Scientific Reports*, 7(1):13282, 2017.

[415] K. Gopalakrishnan, S. K. Khaitan, A. Choudhary, and A. Agrawal. Deep convolutional neural networks with transfer learning for computer vision-based data-driven pavement distress detection. *Construction and Building Materials*, 157:322–330, 2017.

[416] K. Gopalakrishnan, H. Gholami, A. Vidyadharan, A. Choudhary, and A. Agrawal. Crack damage detection in unmanned aerial vehicle images of civil infrastructure using pre-trained deep learning model. *International Journal for Traffic and Transport Engineering*, 8(1):1–14, 2018.

[417] K. Choudhary, K. F. Garrity, C. Camp, S. V. Kalinin, R. Vasudevan, M. Ziatdinov, and F. Tavazza. Computational scanning tunneling microscope image database. *Scientific Data*, 8(1):57, 2021.

[418] K. Kaufmann, C. Zhu, A. S. Rosengarten, and K. S. Vecchio. Deep neural network enabled space group identification in ebsd. *Microscopy and Microanalysis*, 26(3):447–457, 2020.

[419] Z. Yang, T. Watari, D. Ichigozaki, K. Morohoshi, Y. Suga, W.-k. Liao, A. Choudhary, and A. Agrawal. Data-driven insights from predictive analytics on heterogeneous experimental data of industrial magnetic materials. In *2019 International Conference on Data Mining Workshops (ICDMW)*, pages 806–813. IEEE, 2019.

[420] Z. Yang, T. Watari, D. Ichigozaki, A. Mitsutoshi, H. Takahashi, Y. Suga, W.-k. Liao, A. Choudhary, and A. Agrawal. Heterogeneous feature fusion based machine learning on shallow-wide and heterogeneous-sparse industrial datasets. In *Pattern Recognition. ICPR International Workshops and Challenges: Virtual Event, January 10–15, 2021, Proceedings, Part IV*, pages 566–577. Springer, 2021.

[421] R. Liu, A. Agrawal, W.-k. Liao, A. Choudhary, and M. De Graef. Materials discovery: understanding polycrystals from large-scale electron patterns. In *2016 IEEE International Conference on Big Data (Big Data)*, pages 2261–2269. IEEE, 2016.

[422] D. Jha, S. Singh, R. Al-Bahrani, W.-k. Liao, A. Choudhary, M. De Graef, and A. Agrawal. Extracting grain orientations from ebsd patterns of polycrystalline materials using convolutional neural networks. *Microscopy and Microanalysis*, 24(5):497–502, 2018.

[423] Z. Yang, S. Papanikolaou, A. C. Reid, W.-k. Liao, A. N. Choudhary, C. Campbell, and A. Agrawal. Learning to predict crystal plasticity at the nanoscale: deep residual networks and size effects in uniaxial compression discrete dislocation simulations. *Scientific Reports*, 10(1):8262, 2020.

[424] Z. Yang, Y. C. Yabansu, D. Jha, W.-k. Liao, A. N. Choudhary, S. R. Kalidindi, and A. Agrawal. Establishing structure-property localization linkages for elastic deformation of three-dimensional high contrast composites using deep learning approaches. *Acta Materialia*, 166:335–345, 2019.

[425] Z. Yang, Y. C. Yabansu, R. Al-Bahrani, W.-k. Liao, A. N. Choudhary, S. R. Kalidindi, and A. Agrawal. Deep learning approaches for mining structure-property linkages in high contrast composites from simulation datasets. *Computational Materials Science*, 151:278–287, 2018.

[426] A. Cecen, H. Dai, Y. C. Yabansu, S. R. Kalidindi, and L. Song. Material structure-property linkages using three-dimensional convolutional neural networks. *Acta Materialia*, 146:76–84, 2018.

[427] J. Madsen, P. Liu, J. Kling, J. B. Wagner, T. W. Hansen, O. Winther, and J. Schiøtz. A deep learning approach to identify local structures in atomic-resolution transmission electron microscopy images. *Advanced Theory and Simulations*, 1(8):1800037, 2018.

[428] A. Maksov, O. Dyck, K. Wang, K. Xiao, D. B. Geohegan, B. G. Sumpter, R. K. Vasudevan, S. Jesse, S. V. Kalinin, and M. Ziatdinov. Deep learning analysis of defect and phase evolution during electron beam-induced transformations in ws2. *npj Computational Materials*, 5(1):12, 2019.

[429] S.-H. Yang, W. Choi, B. W. Cho, F. O.-T. Agyapong-Fordjour, S. Park, S. J. Yun, H.-J. Kim, Y.-K. Han, Y. H. Lee, K. K. Kim, et al. Deep learning-assisted quantification of atomic dopants and defects in 2d materials. *Advanced Science*, 8(16):2101099, 2021.

[430] G. Roberts, S. Y. Haile, R. Sainju, D. J. Edwards, B. Hutchinson, and Y. Zhu. Deep learning for semantic segmentation of defects in advanced stem images of steels. *Scientific Reports*, 9(1):12744, 2019.

[431] W. Li, K. G. Field, and D. Morgan. Automated defect analysis in electron microscopic images. *npj Computational Materials*, 4(1):36, 2018.

[432] O. S. Ovchinnikov, A. O'Hara, S. Jesse, B. M. Hudak, S.-Z. Yang, A. R. Lupini, M. F. Chisholm, W. Zhou, S. V. Kalinin, A. Y. Borisevich, et al. Detection of defects in atomic-resolution images of materials using cycle analysis. *Advanced Structural and Chemical Imaging*, 6(1):1–9, 2020.

[433] M. Ziatdinov, A. Maksov, and S. V. Kalinin. Learning surface molecular structures via machine vision. *npj Computational Materials*, 3(1):31, 2017.

[434] L. Vlcek, M. Ziatdinov, A. Maksov, A. Tselev, A. P. Baddorf, S. V. Kalinin, and R. K. Vasudevan. Learning from imperfections: predicting structure and thermodynamics from atomic imaging of fluctuations. *ACS Nano*, 13(1):718–727, 2019.

[435] R. Cohn, I. Anderson, T. Prost, J. Tiarks, E. White, and E. Holm. Instance segmentation for direct measurements of satellites in metal powders and automated microstructural characterization from image data. *JOM*, 73(7):2159–2172, 2021.

[436] B. L. DeCost, M. D. Hecht, T. Francis, B. A. Webler, Y. N. Picard, and E. A. Holm. Uhcsdb: ultrahigh carbon steel micrograph database: tools for exploring large heterogeneous microstructure datasets. *Integrating Materials and Manufacturing Innovation*, 6:197–205, 2017.

[437] B. L. DeCost, B. Lei, T. Francis, and E. A. Holm. High throughput quantitative metallography for complex microstructures using deep learning: a case study in ultrahigh carbon steel. *Microscopy and Microanalysis*, 25(1):21–29, 2019.

[438] T. Stan, Z. T. Thompson, and P. W. Voorhees. Optimizing convolutional neural networks to perform semantic segmentation on large materials imaging datasets: x-ray tomography and serial sectioning. *Materials Characterization*, 160:110119, 2020.

[439] Z. Jiang, J. Li, Y. Yang, L. Mu, C. Wei, X. Yu, P. Pianetta, K. Zhao, P. Cloetens, F. Lin, et al. Machine-learning-revealed statistics of the particle-carbon/binder detachment in lithium-ion battery cathodes. *Nature Communications*, 11(1):2310, 2020.

[440] K. de Haan, Z. S. Ballard, Y. Rivenson, Y. Wu, and A. Ozcan. Resolution enhancement in scanning electron microscopy using deep learning. *Scientific Reports*, 9(1):1–7, 2019.

[441] J. M. Ede and R. Beanland. Partial scanning transmission electron microscopy with deep learning. *Scientific Reports*, 10(1):8332, 2020.

[442] O. Furat, D. P. Finegan, Z. Yang, T. Kirstein, K. Smith, and V. Schmidt. Super-resolving microscopy images of li-ion electrodes for fine-feature quantification using generative adversarial networks. *npj Computational Materials*, 8(1):68, 2022.

[443] Z. Liu, T. Bicer, R. Kettimuthu, D. Gursoy, F. De Carlo, and I. Foster. Tomogan: low-dose synchrotron x-ray tomography with generative adversarial networks. *Journal of the Optical Society of America A*, 37(3):422–434, 2020.

[444] S. Müller, C. Sauter, R. Shunmugasundaram, N. Wenzler, V. De Andrade, F. De Carlo, E. Konukoglu, and V. Wood. Deep learning-based segmentation of lithium-ion battery microstructures enhanced by artificially generated electrodes. *Nature Communications*, 12(1):6205, 2021.

[445] W. Ma, E. J. Kautz, A. Baskaran, A. Chowdhury, V. Joshi, B. Yener, and D. J. Lewis. Image-driven discriminative and generative machine learning algorithms for establishing microstructure–processing relationships. *Journal of Applied Physics*, 128(13):134901, 2020.

[446] I. Goytom, Q. Wang, T. Yu, K. Dai, K. Sankaran, X. Zhou, and D. Lin. Nanoscale microscopy images colorization using neural networks. *arXiv preprint arXiv:1912.07964*, 2019.

[447] J. Na, G. Kim, S.-H. Kang, S.-J. Kim, and S. Lee. Deep learning-based discriminative refocusing of scanning electron microscopy images for materials science. *Acta Materialia*, 214:116987, 2021.

[448] M. Rashidi and R. A. Wolkow. Autonomous scanning probe microscopy in situ tip conditioning through machine learning. *ACS Nano*, 12(6):5185–5189, 2018.

[449] L. Scime, D. Siddel, S. Baird, and V. Paquit. Layer-wise anomaly detection and classification for powder bed additive manufacturing processes: a machine-agnostic algorithm for real-time pixel-wise semantic segmentation. *Additive Manufacturing*, 36:101453, 2020.

[450] S. Eppel, H. Xu, M. Bismuth, and A. Aspuru-Guzik. Computer vision for recognition of materials and vessels in chemistry lab settings and the vector-labpics data set. *ACS Central Science*, 6(10):1743–1752, 2020.

[451] W. W. Fleuren and W. Alkema. Application of text mining in the biomedical domain. *Methods*, 74:97–106, 2015.

[452] L. Hawizy, D. M. Jessop, N. Adams, and P. Murray-Rust. Chemicaltagger: a tool for semantic text-mining in chemistry. *Journal of Cheminformatics*, 3:1–13, 2011.

[453] O. Kononova, H. Huo, T. He, Z. Rong, T. Botari, W. Sun, V. Tshitoyan, and G. Ceder. Text-mined dataset of inorganic materials synthesis recipes. *Scientific Data*, 6(1):203, 2019.

[454] T. He, W. Sun, H. Huo, O. Kononova, Z. Rong, V. Tshitoyan, T. Botari, and G. Ceder. Similarity of precursors in solid-state synthesis as text-mined from scientific literature. *Chemistry of Materials*, 32(18):7861–7873, 2020.

[455] C. J. Court and J. M. Cole. Auto-generated materials database of curie and néel temperatures via semi-supervised relationship extraction. *Scientific Data*, 5(1):1–12, 2018.

[456] S. Huang and J. M. Cole. A database of battery materials auto-generated using chemdataextractor. *Scientific Data*, 7(1):260, 2020.

[457] E. J. Beard, G. Sivaraman, Á. Vázquez-Mayagoitia, V. Vishwanath, and J. M. Cole. Comparative dataset of experimental and computational attributes of uv/vis absorption spectra. *Scientific Data*, 6(1):307, 2019.

[458] O. Tayfuroglu, A. Kocak, and Y. Zorlu. In silico investigation into h2 uptake in mofs: combined text/data mining and structural calculations. *Langmuir*, 36(1):119–129, 2019.

[459] L. Weston, V. Tshitoyan, J. Dagdelen, O. Kononova, A. Trewartha, K. A. Persson, G. Ceder, and A. Jain. Named entity recognition and normalization applied to large-scale information extraction from the materials science literature. *Journal of Chemical Information and Modeling*, 59(9):3692–3702, 2019.

[460] A. C. Vaucher, F. Zipoli, J. Geluykens, V. H. Nair, P. Schwaller, and T. Laino. Automated extraction of chemical synthesis actions from experimental procedures. *Nature Communications*, 11(1):3601, 2020.

[461] E. Kim, K. Huang, S. Jegelka, and E. Olivetti. Virtual screening of inorganic materials synthesis parameters with deep learning. *npj Computational Materials*, 3(1):53, 2017.

[462] E. Kim, Z. Jensen, A. van Grootel, K. Huang, M. Staib, S. Mysore, H.-S. Chang, E. Strubell, A. McCallum, S. Jegelka, et al. Inorganic materials synthesis planning with literature-trained neural networks. *Journal of Chemical Information and Modeling*, 60(3):1194–1201, 2020.

[463] P. B. d. Castro, K. Terashima, T. D. Yamamoto, Z. Hou, S. Iwasaki, R. Matsumoto, S. Adachi, Y. Saito, P. Song, H. Takeya, et al. Machine-learning-guided discovery of the gigantic magnetocaloric effect in hob2 near the hydrogen liquefaction temperature. *NPG Asia Materials*, 12(1):35, 2020.

[464] C. B. Cooper, E. J. Beard, Á. Vázquez-Mayagoitia, L. Stan, G. B. Stenning, D. W. Nye, J. A. Vigil, T. Tomar, J. Jia, G. B. Bodedla, et al. Design-to-device approach affords panchromatic co-sensitized solar cells. *Advanced Energy Materials*, 9(5):1802820, 2019.

[465] D. Ho, A. S. Shkolnik, N. J. Ferraro, B. A. Rizkin, and R. L. Hartman. Using word embeddings in abstracts to accelerate metallocene catalysis polymerization research. *Computers & Chemical Engineering*, 141:107026, 2020.

[466] W. X. Zhao, K. Zhou, J. Li, T. Tang, X. Wang, Y. Hou, Y. Min, B. Zhang, J. Zhang, Z. Dong, et al. A survey of large language models. *arXiv preprint arXiv:2303.18223*, 2023.

[467] A. Dunn, J. Dagdelen, N. Walker, S. Lee, A. S. Rosen, G. Ceder, K. Persson, and A. Jain. Structured information extraction from complex scientific text with fine-tuned large language models. *arXiv preprint arXiv:2212.05238*, 2022.

[468] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[469] T. Gupta, M. Zaki, N. A. Krishnan, and Mausam. Matscibert: a materials domain language model for text mining and information extraction. *npj Computational Materials*, 8(1):102, 2022.

[470] S. Huang and J. M. Cole. Batterydataextractor: battery-aware text-mining software embedded with bert models. *Chemical Science*, 13(39):11487–11495, 2022.

[471] P. Shetty, A. C. Rajan, C. Kuenneth, S. Gupta, L. P. Panchumarti, L. Holm, C. Zhang, and R. Ramprasad. A general-purpose material property data extraction pipeline from large polymer corpora using natural language processing. *npj Computational Materials*, 9(1):52, 2023.

[472] D. A. Boiko, R. MacKnight, and G. Gomes. Emergent autonomous scientific research capabilities of large language models. *arXiv preprint arXiv:2304.05332*, 2023.

[473] T. Xie, Y. Wa, W. Huang, Y. Zhou, Y. Liu, Q. Linghu, S. Wang, C. Kit, C. Grazian, and B. Hoex. Large language models as master key: unlocking the secrets of materials science with gpt. *arXiv preprint arXiv:2304.02213*, 2023.

[474] A. M. Bran, S. Cox, A. D. White, and P. Schwaller. Chemcrow: augmenting large-language models with chemistry tools. *arXiv preprint arXiv:2304.05376*, 2023.

[475] Y. Zhao, N. Schiffmann, A. Koeppe, N. Brandt, E. C. Bucharsky, K. G. Schell, M. Selzer, and B. Nestler. Machine learning assisted design of experiments for solid state electrolyte lithium aluminum titanium phosphate. *Frontiers in Materials*, 9:821817, 2022.

[476] J. B. Goodenough and Y. Kim. Challenges for rechargeable Li batteries. *Chemistry of Materials*, 22(3):587–603, 2010.

[477] V. Aravindan, J. Gnanaraj, S. Madhavi, and H.-K. Liu. Lithium-ion conducting electrolyte salts for lithium batteries. *Chemistry–A European Journal*, 17(51):14326–14346, 2011.

[478] A. Manthiram, X. Yu, and S. Wang. Lithium battery chemistries enabled by solid-state electrolytes. *Nature Reviews Materials*, 2(4):1–16, 2017.

[479] H. Aono, E. Sugimoto, Y. Sadaoka, N. Imanaka, and G.-y. Adachi. Ionic conductivity and sinterability of lithium titanium phosphate system. *Solid State Ionics*, 40:38–42, 1990.

[480] A. Rossbach, F. Tietz, and S. Grieshammer. Structural and transport properties of lithium-conducting nasicon materials. *Journal of Power Sources*, 391:1–9, 2018.

[481] Q. Ma, Q. Xu, C.-L. Tsai, F. Tietz, and O. Guillon. A novel sol–gel method for large-scale production of nanopowders: preparation of $Li_{1.5}Al_{0.5}Ti_{1.5}(PO_4)_3$ as an example. *Journal of the American Ceramic Society*, 99(2):410–414, 2016.

[482] M. Pérez-Estébanez, J. Isasi-Marín, D. Többens, A. Rivera-Calzada, and C. León. A systematic study of nasicon-type $Li_{1+x}M_xTi_{2\text{-}x}(PO_4)_3$ (M: Cr, Al, Fe) by neutron diffraction and impedance spectroscopy. *Solid State Ionics*, 266:1–8, 2014.

[483] J. Narváez-Semanate and A. Rodrigues. Microstructure and ionic conductivity of $Li_{1+x}Al_xTi_{2\text{-}x}(PO_4)_3$ nasicon glass-ceramics. *Solid State Ionics*, 181(25-26):1197–1204, 2010.

[484] E. Bucharsky, K. Schell, A. Hintennach, and M. Hoffmann. Preparation and characterization of sol–gel derived high lithium ion conductive nzp-type ceramics $Li_{1+x}Al_xTi_{2\text{-}x}(PO_4)_3$. *Solid State Ionics*, 274:77–82, 2015.

[485] K. Arbi, S. Mandal, J. Rojo, and J. Sanz. Dependence of ionic conductivity on composition of fast ionic conductors $Li_{1+x}Ti_{2\text{-}x}Al_x(PO_4)3$, $0 \leq x \leq 0.7$, a parallel nmr and electric impedance study. *Chemistry of Materials*, 14(3):1091–1097, 2002.

[486] N. Schiffmann, E. C. Bucharsky, K. G. Schell, C. A. Fritsch, M. Knapp, and M. J. Hoffmann. Upscaling of latp synthesis: stoichiometric screening of phase purity and microstructure to ionic conductivity maps. *Ionics*, 27(5):2017–2025, 2021.

[487] T. Hupfer, E. Bucharsky, K. Schell, and M. Hoffmann. Influence of the secondary phase litiopo4 on the properties of $Li_{1+x}Al_xTi_{2\text{-}x}(PO_4)_3$ (x= 0; 0.3). *Solid State Ionics*, 302:49–53, 2017.

[488] Y. Wang, W. D. Richards, S. P. Ong, L. J. Miara, J. C. Kim, Y. Mo, and G. Ceder. Design principles for solid-state lithium superionic conductors. *Nature Materials*, 14(10):1026–1031, 2015.

[489] Y. Zhang, X. He, Z. Chen, Q. Bai, A. M. Nolan, C. A. Roberts, D. Banerjee, T. Matsunaga, Y. Mo, and C. Ling. Unsupervised discovery of solid-state lithium ion conductors. *Nature Communications*, 10(1):1–7, 2019.

[490] Z. Ahmad, T. Xie, C. Maheshwari, J. C. Grossman, and V. Viswanathan. Machine learning enabled computational screening of inorganic solid electrolytes for suppression of dendrite formation in lithium metal anodes. *ACS Central Science*, 4(8):996–1006, 2018.

[491] K. Fujimura, A. Seko, Y. Koyama, A. Kuwabara, I. Kishida, K. Shitara, C. A. Fisher, H. Moriwake, and I. Tanaka. Accelerated materials design of lithium superionic conductors based on first-principles calculations and machine learning algorithms. *Advanced Energy Materials*, 3(8):980–985, 2013.

[492] 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.

[493] N. Cressie. The origins of kriging. *Mathematical Geology*, 22(3):239–252, 1990.

[494] D. H. Wolpert and W. G. Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82, 1997.

[495] T. Hupfer, E. C. Bucharsky, K. G. Schell, A. Senyshyn, M. Monchak, M. J. Hoffmann, and H. Ehrenberg. Evolution of microstructure and its relation to ionic conductivity in $Li_{1+x}Al_xTi_{2\text{-}x}(PO_4)_3$. *Solid State Ionics*, 288:235–239, 2016.

[496] S. D. Jackman and R. A. Cutler. Effect of microcracking on ionic conductivity in latp. *Journal of Power Sources*, 218:65–72, 2012.

[497] K. Waetzig, A. Rost, U. Langklotz, B. Matthey, and J. Schilm. An explanation of the microcrack formation in $Li_{1.3}Al_{0.3}Ti_{1.7}(PO_4)_3$ ceramics. *Journal of the European Ceramic Society*, 36(8):1995–2001, 2016.

[498] M. N. Rahaman. *Sintering of ceramics*. CRC press, 2007.

[499] Y. Zhao, P. Altschuh, J. Santoki, L. Griem, G. Tosato, M. Selzer, A. Koeppe, and B. Nestler. Characterization of porous membranes using artificial neural networks. *Acta Materialia*, 253:118922, 2023.

[500] P. Altschuh, Y. C. Yabansu, J. Hötzer, M. Selzer, B. Nestler, and S. R. Kalidindi. Data science approaches for microstructure quantification and feature identification in porous membranes. *Journal of Membrane Science*, 540:88–97, 2017.

[501] G. E. Fridley, C. A. Holstein, S. B. Oza, and P. Yager. The evolution of nitrocellulose as a material for bioassays. *MRS Bulletin*, 38(4):326–330, 2013.

[502] C. De Boor and C. De Boor. *A practical guide to splines*, volume 27. Springer-Verlag, 1978.

[503] G. Martínez-Criado, J. Villanova, R. Tucoulou, D. Salomon, J.-P. Suuronen, S. Labouré, C. Guilloud, V. Valls, R. Barrett, E. Gagliardini, et al. Id16b: a hard x-ray nanoprobe beamline at the esrf for nano-analysis. *Journal of Synchrotron Radiation*, 23(1):344–352, 2016.

[504] P. Altschuh, W. Kunz, M. Bremerich, A. Reiter, M. Selzer, and B. Nestler. Wicking in porous polymeric membranes: determination of an effective capillary radius to predict the flow behavior in lateral flow assays. *Membranes*, 12(7):638, 2022.

[505] P. Altschuh. *Skalenübergreifende Analyse makroporöser Membranen im Kontext digitaler Zwillinge*. PhD thesis, Karlsruher Institut für Technologie (KIT), 2020.

[506] K. Weiss, T. M. Khoshgoftaar, and D. Wang. A survey of transfer learning. *Journal of Big Data*, 3(1):1–40, 2016.

[507] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics*, pages 249–256. JMLR Workshop and Conference Proceedings, 2010.

[508] D. P. Kingma and J. Ba. Adam: a method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[509] Y. Zhao, S.-K. Otto, T. Lombardo, A. Henss, A. Koeppe, M. Selzer, J. Janek, and B. Nestler. Identification of lithium compounds on surfaces of lithium metal anode with machine-learning-assisted analysis of tof-sims spectra. *ACS Applied Materials & Interfaces*, 15(43):50469–50478, 2023.

[510] J. E. Jackson. Principal components and factor analysis: part i—principal components. *Journal of Quality Technology*, 12(4):201–213, 1980.

[511] J. Lee, I. Gilmore, I. Fletcher, and M. Seah. Multivariate image analysis strategies for tof-sims images with topography. *Surface and Interface Analysis*, 41(8):653–665, 2009.

[512] J. Lee, I. Gilmore, and M. Seah. Quantification and methodology issues in multivariate analysis of tof-sims data for mixed organic systems. *Surface and Interface Analysis*, 40(1):1–14, 2008.

[513] D. J. Graham, M. S. Wagner, and D. G. Castner. Information from complexity: challenges of tof-sims data interpretation. *Applied Surface Science*, 252(19):6860–6868, 2006.

[514] D. M. Powers. Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation. *arXiv preprint arXiv:2010.16061*, 2020.

[515] E. Fix and J. L. Hodges Jr. Discriminatory analysis-nonparametric discrimination: small sample performance. Technical report, California Univ Berkeley, 1952.

[516] D. J. Hand and K. Yu. Idiot's bayes—not so stupid after all? *International Statistical Review*, 69(3):385–398, 2001.

[517] K. Pearson. Vii. note on regression and inheritance in the case of two parents. *Proceedings of the Royal Society of London*, 58(347-352):240–242, 1895.

[518] G. F. Trindade, M.-L. Abel, C. Lowe, R. Tshulu, and J. F. Watts. A time-of-flight secondary ion mass spectrometry/multivariate analysis (tof-sims/mva) approach to identify phase segregation in blends of incompatible but extremely similar resins. *Analytical Chemistry*, 90(6):3936–3941, 2018.

[519] P. Visser, K. Marcoen, G. Trindade, M. Abel, J. Watts, T. Hauffman, J. Mol, and H. Terryn. The chemical throwing power of lithium-based inhibitors from organic coatings on aa2024-t3. *Corrosion Science*, 150:194–206, 2019.

[520] G. A. Seber and A. J. Lee. *Linear regression analysis*, volume 330. John Wiley & Sons, 2003.

[521] D. A. Freedman. *Statistical models: theory and practice.* Cambridge University Press, 2009.

[522] D. G. Krige. A statistical approach to some basic mine valuation problems on the witwatersrand. *Journal of the Southern African Institute of Mining and Metallurgy*, 52(6):119–139, 1951.

[523] C. Williams and C. Rasmussen. Gaussian processes for regression. *Advances in Neural Information Processing Systems*, 8, 1995.

[524] A. G. Shard, A. Miisho, J.-L. Vorng, R. Havelund, I. S. Gilmore, and S. Aoyagi. A two-point calibration method for quantifying organic binary mixtures using secondary ion mass spectrometry in the presence of matrix effects. *Surface and Interface Analysis*, 54(4):363–373, 2022.

[525] S. Aoyagi and K. Matsuda. Quantitative analysis of tof-sims data of a two organic compound mixture using an autoencoder and simple artificial neural networks. *Rapid Communications in Mass Spectrometry*, 37(4):e9445, 2023.

[526] J. Hill, G. Mulholland, K. Persson, R. Seshadri, C. Wolverton, and B. Meredig. Materials science with large-scale data and informatics: unlocking new opportunities. *MRS Bulletin*, 41(5):399–409, 2016.

[527] N. Brandt, L. Griem, C. Herrmann, E. Schoof, G. Tosato, Y. Zhao, P. Zschumme, and M. Selzer. Kadi4mat: a research data infrastructure for materials science. *Data Science Journal*, 20:8–8, 2021.

[528] L. Griem, P. Zschumme, M. Laqua, N. Brandt, E. Schoof, P. Altschuh, and M. Selzer. Kadistudio: fair modelling of scientific research processes. *Data Science Journal*, 21:16–16, 2022.

[529] M. D. Wilkinson, M. Dumontier, I. J. Aalbersberg, G. Appleton, M. Axton, A. Baak, N. Blomberg, J.-W. Boiten, L. B. da Silva Santos, P. E. Bourne, et al. The fair guiding principles for scientific data management and stewardship. *Scientific Data*, 3(1):1–9, 2016.

[530] S. Zhao, J. Song, and S. Ermon. Infovae: information maximizing variational autoencoders. *arXiv preprint arXiv:1706.02262*, 2017.

[531] J. Santoki, D. Schneider, M. Selzer, F. Wang, M. Kamlah, and B. Nestler. Phase-field study of surface irregularities of a cathode particle during intercalation. *Modelling and Simulation in Materials Science and Engineering*, 26(6):065013, 2018.

List of publications

[1]. **Y. Zhao**, S.-K. Otto, N. Brandt, M. Selzer, and B. Nestler. Application of random forests in tof-sims data. *Procedia Computer Science*, 176:410–419, 2020. DOI: 10.1016/j.procs.2020.08.042.

[2]. N. Brandt, L. Griem, C. Herrmann, E. Schoof, G. Tosato, **Y. Zhao**, P. Zschumme, and M. Selzer. Kadi4mat: a research data infrastructure for materials science. *Data Science Journal*, 20:8–8, 2021. DOI: 10.5334/dsj-2021-008.

[3]. **Y. Zhao**, N. Schiffmann, A. Koeppe, N. Brandt, E. C. Bucharsky, K. G. Schell, M. Selzer, and B. Nestler. Machine learning assisted design of experiments for solid state electrolyte latp. *Frontiers in Materials*, 9:821817, 2022. DOI: 10.3389/fmats.2022.821817.

[4]. **Y. Zhao**, P. Altschuh, J. Santoki, L. Griem, G. Tosato, M. Selzer, A. Koeppe, and B. Nestler. Characterization of porous membranes using artificial neural networks. *Acta Materialia*, 253, 118922, 2023. DOI: 10.1016/j.actamat.2023.118922.

[5]. **Y. Zhao**, S.-K. Otto, T. Lombardo, A. Henss, A. Koeppe, M. Selzer, J. Janek, and B. Nestler. Identification of lithium compounds on surfaces of lithium metal anode with machine-learning-assisted analysis of tof-sims spectra. *ACS Applied Materials & Interfaces*, 15(43):50469–50478, 2023. DOI: 10.1021/acsami.3c09643.

[6]. Y. Ji, A. Koeppe, P. Altschuh, D. Rajagopal, **Y. Zhao**, W. Chen, Y. Zhang, Y. Zheng, and B. Nestler. Towards automatic feature extraction and sample generation of grain structure by variational autoencoder. *Computational Materials Science*, 232:112628, 2024. DOI: 10.1016/j.commatsci.2023.112628.