Machine learning applications in macromolecular X-ray crystallography

Melanie Vollmar and Gwyndaf Evans

Diamond Light Source Ltd., Harwell Science & Innovation Campus, Didcot, UK; Rosalind Franklin Institute, Harwell Science & Innovation Campus, Didcot, UK

ABSTRACT
After more than half a century of evolution, machine learning and artificial intelligence, in general, are entering a truly exciting era of broad application in commercial and research sectors. In X-ray crystallography, and its application to structural biology, machine learning is finding a home within expert and automated systems, is forecasting experiment and data analysis outcomes, is predicting whether crystals can be grown and even generating macromolecular structures. This review provides a historical perspective on AI and machine learning, offers an introduction and guide to its application in crystallography and concludes with topical examples of how it is currently influencing macromolecular crystallography.

ARTICLE HISTORY
Received 7 May 2021
Accepted 16 September 2021

KEYWORDS
Machine learning; big data; automation; macromolecular X-ray crystallography; synchrotron; structural biology

Contents

Introduction 55

Summary and evolution of machine learning and artificial intelligence 58

Common concepts and terminology 62

Supervised, unsupervised and reinforcement learning 62
Regression and classification 62

Types of algorithms 63
Dimensionality reduction 63
Bayesian algorithms 63
Shallow learners 65
Deep learners 66

Exploratory data analysis and initial assessment 67
Find attributes/features/variables in the dataset 69
Conduct univariate data analysis to identify data distributions 70
Use bivariate/multivariate data analysis to identify relationships and interactions 70
Detect missing values 71
Detect outliers 71
Feature engineering 71

CONTACT
Melanie Vollmar melanie.vollmar@diamond.ac.uk

© 2021 The Author(s), Published by Informa UK Limited, trading as Taylor & Francis Group
This is an Open Access article distributed under the terms of the Creative Commons Attribution License (http://creativecommons.org/licenses/by/4.0/), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.
Introduction

After more than half a century in its development with numerous false starts along the way, artificial intelligence (AI) and its sub-field machine learning (ML; not to be confused with maximum likelihood, a method frequently employed in macromolecular crystallography; MX) are now beginning to have a measurable impact in MX. Real-world examples for machine learning and AI applications that many of us have become familiar with are driverless cars, speech recognition and targeted advertisement. These systems are very much of a dynamic nature requiring near real-time decision making and adaptation on different levels within the application. In MX, many automated processes are static with a defined start and end point running in a linear fashion. Many such processes may run in parallel and communication between them is typically limited. Key decision-making points in automation provide excellent opportunities to apply ML and AI-based control and guidance using the experiences gained from real-world applications like those listed above. For MX, several factors have contributed to a burst of activity and relative success: first, there is greater accessibility to software libraries that allow numerous approaches to machine learning to be tried and evaluated for MX applications; second, there is a wealth of training data available that were the result of decades of macromolecular crystal structure determinations, predominantly as a by-product of the structural genomics boom in the early 2000s [1–5]; and finally, the wide availability to many researchers of high-performance computing facilities makes the management of these data and training of machine learning systems a more easily achievable task.

The last 20 years have witnessed a remarkable revolution in synchrotron macromolecular X-ray crystallography. Broad application of instrument automation, and expert control and analysis systems have conspired to make MX accessible to a far wider pool of structural and medical biologists, both academia and industry based, who strive to understand the
Figure 1. Cumulative number of protein structure depositions since the start of the Protein Data Bank. Plotted are the contributing methods X-ray diffraction using synchrotron radiation (X-ray synchrotron; source: https://biosync.rcsb.org/) or using other sources (X-ray other), nuclear magnetic resonance (NMR) and cryo-electron microscopy (EM) by year.

fundamental mechanisms of biological molecules and cells, the basis of diseases and seek to develop drugs and therapeutics to tackle them.

Around 88% of the ∼180,000 MX structures deposited in the PDB (Protein Data Bank) [6] have come from synchrotron data with currently about 10,000 structures per year being deposited (Figure 1). This has been enabled by automation of X-ray beamlines, logistics to manage sample shipping and handling at the synchrotron, fully automated [7] or unattended data collection (UDC) [8,9], automated data analysis pipelines [10] and underpinning it all a wealth of investment in X-ray beamline sources, optics, detectors, end-stations [11] and expert staff to develop and maintain these complex systems. The COVID-19 pandemic has forced the hand of many synchrotrons in making their beamlines available only by remote access. Fortunately, the developments at many synchrotron sites worldwide over the last two decades have positioned MX perfectly to cope with the practical constraints imposed by the pandemic. For example, at Diamond throughout 2020 user access to MX beamlines was almost exclusively remote and in the 11 months from June 2020 over 33,000 data sets were measured through UDC, with typically less than 3 min being used for each crystal sample. This is from over 500 UDC user sessions that were run alongside more typical remote access sessions where users interactively measured their data. Automated data collection and analysis at MX beamlines is now seen as an essential component of current and future operations for many synchrotrons.

The automated analysis of the wealth of data arising from modern MX beamlines, as illustrated above, presents a whole new set of challenges. In recent years, a rather straightforward brute-force approach to automated data analysis has been challenged by the evolution of beamline technologies resulting in extraordinary data rates and data volumes from the latest high frame rate (> 100 fps) and multi-megapixel detectors (typically between 4 and 16 Mpixels per frame). The challenge is compounded by the desire by many users, and indeed the automated systems themselves to have real-time feedback on the
quality of data being measured. With the emerging minimization of human intervention from data assessment and analysis, there has been an increase in the importance of automated expert systems to steer such analysis. More fundamentally, as the yearly quantities of measured data from large-scale facilities like Diamond reach many-Petabyte levels the question of whether ML and/or AI applications can be used to determine which data to store and which to discard is being asked [12].

Many expert systems to date have represented linear sequences of tasks that require little decision making but either fail or succeed based on the data quality [13,14]. However, the evaluation of crystallographic data quality is a contextual and multidimensional problem with no single metric telling the full story about the usefulness of a data set in answering a given scientific question. Trained crystallographers use their experience and knowledge, together with multiple quality indicators [15–17], to make decisions about which software to use, which structure determination method is likely to succeed, whether it is worth proceeding with analysis or whether it may be better to return to the experimental stage of data collection or even sample preparation [18]. But coding this expertise and knowledge into a comprehensive, robust and helpful autonomous data analysis pipeline is non-trivial.

Machine learning and artificial intelligence are, however, beginning to play an important role in addressing this problem by adopting the role of an expert crystallographer. Indeed, structure determination by crystallography is a problem that is particularly well-suited to the application of machine learning and is a problem that stands to benefit substantially from it. The wealth of existing diffraction data and solved structures, captured for macromolecular crystallography in the PDB and the many data sets stored long term at synchrotron sites, provide a rich foundation for ML and AI methods. As well as assisting in the automation of standard structure solution methods, even more challenging structure problems in crystallography and structural biology around protein structure prediction from amino acid sequence data [19], the question of crystallizability and even the detection of the presence of crystals in crystallization trials [20] can now be tackled. Bruno et al. [20] estimate that for commonly used experimental setups in a high-throughput environment (often around 1,000 96-well plates per year) about 1 hour will be needed to manually evaluate the crystallization trials in one such plate. Repeat evaluations are usually done throughout the lifespan of such a trial. Assuming 1 hour/plate, it would take 42 full days to evaluate each of the 1,000 plates once. Five repeats will amount to 210 24-hour days. Although not stated directly by Bruno et al. [20] one imagines that a deep learning-based system would be able to achieve the same task in a much shorter time.

For the first time in decades, there is considerable, and realistic, interest about the practical applicability and usefulness of such methods and the results they produce. Recently, this interest has reached fever-pitch with the latest results of the biennial Critical Assessment of protein Structure Prediction (CASP) [21] that challenges scientist to solve the 50-year-old grand challenge of protein folding. Remarkable results from the use of AlphaFold2 have offered a clearer illustration of the potential of AI for tackling one of the most complex problems in biology.

This review offers a crystallographer’s eye view of artificial intelligence and machine learning, starting with a comprehensive summary of their evolution, then recipes for their application in crystallography and finishing with recent examples of their successful utilization in our field.
Summary and evolution of machine learning and artificial intelligence

Statistical methods are the basis for the development of algorithms that form the core of machine learning applications within the field of artificial intelligence. The relationship between artificial intelligence, machine learning and deep learning is given in Figure 2. One key concept in artificial intelligence is that of not explicitly programming an algorithm, i.e. a developer or programmer will not directly encode the prediction answer for each possible sample. Instead, a statistical model with a set of parameters (weights) representing general decision/association rules is created based on the known properties of a subset of data. The learned weights can then be applied to any new sample and a prediction about this sample is made. Any information or pattern found within the data distribution is then used in decision making after having been discovered by the algorithm itself [22,23]. Limitations are given by the design of the algorithm to be trained and discovery of new knowledge is unlikely as they generally lack human ingenuity and cross-disciplinary insight. Instead, ML and AI-based applications serve as tools to work with large data volumes, highly specific tasks and automated decision making to speed up processes. No single person can be credited with the invention of ML and/or AI. Instead, this has been an evolutionary process over many decades with uncounted known and unknown developers being involved. Below we give some key events that pushed these developments. A graphical timeline is given in Figure 3.

The underlying idea, for artificial intelligence in particular, is based on a model of the way brain cells (neurons) or nodes in a neural network communicate with each other and transfer information through electrical signals as was proposed by McCulloch and Pitts [24] and Hebb [25]. Special emphasis is placed on the visual cortex based on the research by Hubel and Wiesel [26,27]. An electrical signal can have an activating or
Figure 3. Timeline machine learning developments. A timeline to show the major key development moments within the last about 80 years.
deactivating effect on a neuron which in turn corresponds to a positive or negative weight in a computational neural network.

John McCarthy, Alan Turing, Marvin Minsky, Allen Newell and Herbert A. Simon are generally considered to be the founding fathers of artificial intelligence. Alan Turing in 1950 [28] devised the Turing Test in which a computer aims to convince a human that it is not a machine but a human. The test requires a computer algorithm to hold a natural language conversation with a human observed by an evaluator. If the evaluator cannot distinguish who in the conversation is a machine and who a human, then the algorithm has successfully passed the test. So far, no AI algorithm developed was successful in the Turing test. In 1952, Marvin Minsky [29] developed the first physical implementation of a neural network with his Stochastic Neuro-Analogue Reinforcement Computer (SNARC). The system was constructed of 40 hardwired vacuum tubes representing neurons and their connections and was able to learn simple concepts through ‘rewards’. John McCarthy, Marvin Minsky, Nathaniel Rochester and Claude E. Shannon wrote a proposal for the Dartmouth Conference in 1956, in which they coined the term ‘artificial intelligence’ [30].

In the 1950s and 60s, Arthur Samuel [31,32] working at IBM developed the first scoring and reward functions to enable a computer to learn and coined the phrase ‘Machine Learning’ and the idea of ‘learning by generalization’ for the first time. In 1957, intrigued by the ideas of connections between neurons and generalized learning, Frank Rosenblatt designed the first perceptron [33]. A simple version of such a perceptron, depicted in Figure 4, is constructed of input nodes randomly connected with each other while their information exchange is altered through weights producing a single output and is the simplest form of a neural network. Each input node (purple) represents a feature or a dendrite in a neuron. Additionally, a constant bias term can be present. Weights can be applied to each feature based on their importance or signal strength in a neuron. All the incoming features and their weights are summed and an activation function is applied. In a binary classification problem, this will be a sigmoid activation function working with ‘AND’ and ‘OR’ statements so that a prediction can only be true or false, ‘1’ or ‘0’. A single output, true or false, is produced, a signal sent to other neurons through the axon, and errors for false predictions are returned to the system via backpropagation. The whole calculation is repeated many times to minimize the error and increase correct predictions.

Cover and Hart [34] are credited for the idea of the ‘nearest neighbour’ rule which set the starting point for pattern recognition algorithms. In most applications, there is no information available about the underlying sample distribution for each class. The class label for each sample is therefore inferred from the labels found for already classified samples in the neighbourhood. As assigning a class label is not influenced by prior knowledge found in the sample distribution but solely done based on what classes are found in proximity, the classification process itself carries an error and is not optimal.

Also, in the 1960s the concept of using multiple layers within a neural network was explored by expanding the simple perceptron by two or more layers. This paved the way for feedforward neural networks [35,36] and the use of backpropagation in the 1970s [37] which allows a neural network to learn from mistakes and adapt to changing situations. Feedforward describes the flow of information from input to output whereas backpropagation is an essential part of deep neural networks as it returns feedback about a system’s performance. Figure 4 gives a very simple idea of how this works in a perceptron.
In the following two decades, the more statistics-based approaches to algorithm design in machine learning separated from the field of artificial intelligence due to lack of progress in the latter. This was largely, as already mentioned above, due to the lack of appropriate computing infrastructure and resources to work with these highly complex systems. Developments in machine learning at the time focused on probability theory and statistics to create practical applications.

A key development to move statistical machine learning forward was the appearance of a series of boosting algorithms. Boosting allows a machine-learning algorithm to reduce the sample bias when learning by combining the results of a series of weak learners into a strong one [38]. Weak learners are algorithms that make predictions based on weak correlations to the ground truth but are still better than random. Combining repetitive weak learning predictors and applying weights, for which different approaches have been developed, to penalize wrong predictions will produce a strong predictive model.

From the 1990s on, mainly due to developments in computational infrastructure, the development of the internet and ever-increasing amounts of data, artificial intelligence had its revival and has been evolving ever since. Artificial neural networks (ANN) are a complex expansion of the simple perceptron capable of learning highly complex tasks. They usually comprise an input and output layer with a series of hidden layers in between. The computational power to detect patterns in given data sits with these hidden layers and their inter- and intra-node connections. Any patterns found in large, multi-dimensional data
are usually too complex to be directly detected by a human whether this is tried through data analysis or in a programmatic way, but are well suited to ANNs of appropriate design. For example, long short-term memory (LSTM) neural networks are currently the basis for speech recognition but were already described in 1997 by Schmidhuber and Hochreiter [39]. LSTMs ‘remember’ events that have happened early on in a timeline and make a connection to the current situation. Their more sophisticated versions are the basis for natural language processing algorithms.

The most recent changes in machine learning and artificial intelligence, which have a direct impact on our lives are facial recognition, self-driving vehicles, human–robot shared workspaces, the Internet of Things, and many areas and new developments in data analysis and health care. With ever-increasing data rates in almost all areas and new developments in algorithms and computing technologies, applications have become increasingly accurate, more efficient and scalable. Ultimately, this has driven machine learning and artificial intelligence applications towards continuous learning.

Common concepts and terminology

Supervised, unsupervised and reinforcement learning

Supervised learning is the dominant learning style in many machine learning applications. It relies on the fact that the label, if a classification of a sample is the desired outcome, or a target value, if regression is the means of prediction, is known for each entity in a training set. Patterns found in the data are then associated with the known result and the algorithm learns this connection. Learning continues until the desired level of accuracy has been reached.

In the case of unsupervised learning, data are given to a system without any additional information and the algorithm detects patterns in the data, which can be used to group or cluster the samples that exhibit similar patterns. The goal is to find rules that group as many samples as possible in as few clusters as possible. After identifying groups/clusters, they can then, for example, be assigned labels and serve as a basis for solving a supervised classification problem. Unsupervised learning is applied in applications using principles like regularization and compression. Semi-supervised learning is a mix of both of the above in that the data is partially labelled and the algorithm has to learn a way of how the unlabelled data can be organized to follow the structure of the samples that already have a solution.

Reinforcement learning is similar to supervised learning as it uses a mapping between input and output during training. The way feedback is provided to the system to improve learning is, however, different from supervised learning. In supervised learning, the correct answer to a mistake is given as feedback, whereas in reinforcement learning a penalty or reward is given. Over time the system, therefore, develops experience through self-learning by maximizing reward and minimizing punishment rather than by being given the correct answers.

Regression and classification

Regression describes a process rather than being the name of an algorithm class or a problem. It describes the relationship between different variables in a dataset and this
connection is refined over multiple iterations while monitoring some measure of error to judge the performance of the model. The output predicted by a regression model is a value on a continuous scale.

Classification is used to predict a label for a sample. A label can be of categorical, often a short-hand description of a class, or numerical nature, where the different descriptive categories are translated into unique numerical values or as one-hot encoding. In one-hot encoding, an 1D array of numbers corresponding to the total length of categories is created, where a particular class is represented as ’1’ and the remaining classes as ’0’. For example, if six class labels are possible, then the one-hot encoding for class ’3’ would be [0. 0. 1. 0. 0. 0.]. This label is used to distinctly identify a sample based on the specific pattern found in the features provided. The output predicted by a classification model is a value on a discrete scale.

**Types of algorithms**

The number of available algorithms to choose from is vast. An introduction to all available forms is beyond the scope of this review. Figure 5 depicts a decision tree which can be utilized to narrow down the type of algorithms one may want to explore for given data. As mentioned in the introduction, machine learning and artificial intelligence are data driven in nature and no assumptions about the underlying distribution of samples should be made. No samples should be excluded or biases introduced by selecting specific subtypes of data unless there are valid reasons, i.e. samples have missing values for some features which cannot be imputed or subsampling is used to explore a newly discovered hypothesis. To explore data from different perspectives it may be necessary to explore a very large number of algorithms.

**Dimensionality reduction**

Dimensionality reduction is an unsupervised method to describe the data with reduced information or dimensions. Dimensionality reduction itself happens at various steps during a training process and, for example, is used to create the latent space in neural networks. Here we use dimensionality reduction in the context of a data exploration step to visualize high dimensional data in a human interpretable way. This is usually done before more in-depth training of algorithms. Dimensionality reduction can be used for classification as well as regression problems. Algorithms used for dimensionality reduction are not restricted to exploratory data analysis but are machine learning applications in their own right. Popular algorithms are ‘Principal Component Analysis’ (PCA) [40], ‘Principal Component Regression’ (PCR), ‘Partial Least Squares Regression’ (PLSR) [41], ‘Multidimensional Scaling’ (MDS) [42,43] and ‘Linear Discriminant Analysis’ (LDA) [44,45]. The use of LDA or PCA is not restricted to dimensionality reduction as one can see from its usage in the automated model building package ARP/wARP [46,47] or when classifying crystallization outcomes [48–51] (see the examples later in this review).

**Bayesian algorithms**

These can be used to look at classification and regression problems alike. Most commonly used are systems applying ‘Naive Bayes’, ‘Gaussian Naive Bayes’, ‘Multinomial Naive
Figure 5. Decision tree to help with algorithm choice. Common situation in which a particular algorithm may be used are explored with the help of a decision tree. The schema can be used as guidance when starting with a new machine learning project.
Bayes’ [52], ‘Bayesian Networks’ (BN) and ‘Bayesian Belief Networks’ (BBN) [53]. They are based on applications of Bayes’ theorem and Bayesian statistics in general [54]. A naïve Bayes classifier has been used to predict the chances of crystallizability for a given protein as will be explained in the example section [55].

**Shallow learners**

Shallow learners can be used to address both classification and regression problems. Below, we give an overview of the types of algorithms available. In a fast-moving field like machine learning, this list is intended to give an overview of well-tested and established systems rather than represent any possible algorithm and it will most likely have any very latest developments missing. These algorithms contain those that are well known within statistics such as ‘Ordinary Least Squares Regression’ (OLSR), ‘Linear Regression’ and ‘Logistic Regression’.

Adding a penalization function for model complexity to certain regression algorithms forces a predictor to be simpler and therefore being better in generalization. This includes for example ‘Ridge Regression’ [56], ‘Least Absolute Shrinkage and Selection Operator’ (LASSO) [57,58] and ‘Elastic Net’. If certain training samples are identified to be crucial for the training of a model then one would like to select an instance-based algorithm such as ‘k-Nearest Neighbour’ (kNN) [59,60] or ‘Support Vector Machines’ (SVM) [61]. SVMs have been used in several applications in protein crystallography. As we will see in the example section, they have been used to predict the chances of successful crystallization of a protein based on its sequence [55,62,63] as well as assessing the outcome of a crystallization trial [51,64]. In the automated model building programme ARP/wARP a SVM is used to identify amino acid side chains following a main chain building step [65]. For all the methods mentioned so far, data normalization is crucial for their operation.

Decision tree algorithms such as the most well-known ‘Classification and Regression Tree’ (CART) [66] or ensembles created of them aim to make predictions based on the actual real values in the data, without any normalization. At each decision point in the tree, a ‘yes/no’ result has to be produced based on the conditions applied at that moment. This can be used for classification and regression problems alike and the algorithms themselves are fast and accurate and hence very widely used in machine learning. Ensemble models often use decision trees as base algorithms and then combine weak predictions of multiple, hundreds to thousands, of trees to create a strong ensemble result. An ensemble classifier will continue to increase the number of trees until all samples in the available data can be classified. Care should be taken as in extreme cases an ensemble will have one tree for each sample meaning it has learned the training data but will be of no use when generalizing, as in the case of a new, unknown sample. Decision tree-based algorithms can be used to solve a classification as well as a regression problem and often apply some form of ‘Boosting’ such as found in ‘Bootstrapped Aggregation’ (Bagging) [67,68], ‘AdaBoost’, ‘Gradient Boosting Machines’ (GBM) [69] and ‘Gradient Boosted Regression Trees’ (GBRT) [70]. ‘Boosting’ is a sequential way of learning, where a weak decision tree relies on information from a previous learning cycle which increases the predictors experience step-by-step over time. ‘Bagging’ allows the trees to learn independently in parallel and combines their experiences for the final prediction. There is also the option to create a ‘Random Forest’ or
'Extreme-Random Forest' [71] in which a very large number of very shallow trees are being combined and various internal parameters in the algorithm are being selected with a level of randomness. Decision trees, AdaBoost and random forests have been used to classify crystallization outcomes [72,73], predict the likelihood of a protein to crystallize [74,75] and provide a decision maker in automated data analysis pipelines [76] (see later examples).

**Deep learners**

The underlying base of state-of-the art deep learning applications is ‘Artificial Neural Network’ (ANN) [77] algorithms. These types of neural networks are designed after the principles found in biological neural networks such as the brain’s visual cortex. ANNs can be used for both regression and classification problems and have far too many variations to list them all. The most well-known algorithms are ‘Perceptron’ [33], ‘Multilayer Perceptron’ (MLP), ‘Back-Propagation’ [78], ‘Stochastic Gradient Descent’ (SGD) and ‘Radial Basis Function Network’ (RBFN). By increasing the complexity of these networks through adding hundreds of layers and having a huge number of connections between the neurons, deep learning algorithms are being created. A small selection of very common and popular deep learning algorithms include: ‘Convolutional Neural Networks’ (CNNs) [36], ‘Recurrent Neural Networks’ (RNNs) [79], ‘Long Short-Term Memory Networks’ (LSTMs) [39], ‘Stacked Auto-Encoders’, ‘Deep Boltzmann Machines’ (DBM) [80] and ‘Deep Belief Networks’ (DBNs) [81]. There are also more specialized algorithms for particular fields for example those concerned with ‘Computer Vision’ (CV), ‘Natural Language Processing’ (NLP), ‘Recommender Systems’, ‘Reinforcement Learning’ [82] and evolutionary algorithms looking at ‘Computational intelligence’. Following the developments and designs of convolutional neural networks, great breakthroughs were achieved with ANNs and AI in other areas. These achievements have now made their way into macromolecular crystallography. The simplest version, a multi-layer perceptron, has been used in an automated model building pipeline [83]. Convolutional neural networks have been trained to predict protein crystallizability [75,84] and classify crystallization outcomes [20,85]. ResNet and more specialized neural network architectures have been used for diffraction image classification [86,87] and protein modelling/structure prediction [88–91]. We will explore these applications in the example section below. In the field of cryo electron microscopy (EM), the software package cryoSPARC [92] uses SGD to identify a set of low-resolution structures based on the particles found in the recorded images. Crucially, SGD does not require an already highly accurate initial model as is normally the case for refinement programs using Bayesian likelihood to solve an optimization problem. The CNN-based YOLO (You Only Look Once) algorithm [93] finds its use in crYOLO [94], another cryo-EM package. Here, YOLO is used to find thousands of particles in cryo-EM micrographs and to draw bounding boxes around them. Finding the particles is the crucial first step to reconstruct a 3D protein structure using cryo-EM data. Another popular cryo-EM package, RELION, on the other hand uses a Bayesian approach to reconstruct a 3D protein model from 2D micrographs [95]. However, as we are focusing on machine learning and AI applications in crystallography, the reader may refer to the citations given above to get more details about implementations in cryo-EM.
Exploratory data analysis and initial assessment

Exploratory data analysis (EDA) is an important process to obtain a good initial grasp of the data one has available and how these data can be used to answer specific questions or hypotheses. These questions and hypotheses are not apparent from the data straight away but can be formulated when carrying out some initial analysis. EDA is often undervalued and neglected since training algorithms and developing predictive models is considered the more interesting challenge. Six basic steps have been defined when carrying out EDA [96]:

1. Find attributes/features/variables in the dataset
2. Conduct univariate data analysis to identify data distributions
3. Use bivariate/multivariate data analysis to identify relationships and interactions between attributes/features/variables
4. Detect missing values and either remove the corresponding samples or find a way of replacing the values (i.e. using the mean or median for these attributes/features/variables)
5. Detect outliers and remove the corresponding sample or set min/max boundaries for the attributes/features/variables
6. Feature engineering to combine attributes/features/variables for dimensionality reduction (see above) and/or to create new attributes/features/variables

Here, we introduce these six basic steps to carry out EDA which help to get an intuition for the data and therefore provide some ideas about how to answer questions and test hypotheses hidden in the data. Additionally, this step is a quality check of the data for its completeness and integrity, identifies gaps in the data, which may require further gathering of samples, and helps to select the appropriate tools and techniques to finally train a predictive model. A small but representative subset of data is sufficient for such analysis and simple summary statistics such as mean, median, maximum and minimum values, first and third quartiles are a good starting point. Higher-order moments such as variance, kurtosis and skew should also be explored. Plotting the data as histograms, line charts, box plots, pairwise scatterplots and correlation matrices is a helpful way to visually explore the data. Such analysis will quickly identify outliers or oddities in the data, e.g. modality of the data distribution, and may reveal correlations, dependencies and redundancies between attributes/features/variables. Tabular data usually have the attributes/features/variables as column headers whereas the rows represent individual samples. Figure 6(a) shows a pair plot matrix where pairs of features are plotted against each other as scatter plots. Samples of the negative class in this binary classification example are given in blue and those belonging to the positive class are coloured orange. Such a set of plots can be used to identify features that have a linear correlation or those that will be suitable to separate groups of data. The diagonal plots the feature on itself and is shown as a density distribution function for each class separately. For feature05 the peaks found in the graphs for the two classes are no longer fully overlapping and may therefore be of high relevance in decision making. Also the scatter for both sample classes are not fully overlapping and have different centre of mass. Calculating Pearson’s correlation coefficient with associated p-values is another way to identify linear correlations between features, see Figure 6(b).
range of correlation coefficients is between +1, a perfect positive correlation (coloured blue here), and −1, a perfect negative correlation (coloured red), with 0 (white) meaning there is no correlation between a pair of features. The diagonal is the self-correlation

Figure 6. Collection of graphs that can be used to visualize the input data and performance results. (a) depicts a pair plot for the eight features used in these examples. All data samples of the training set have been against each other and the colouring is done based on the two possible classes '0' in blue and '1' in orange. (b) is a correlation matrix, a visual representation for Pearson’s correlation coefficients calculated for feature pairs of the training set data. (c)–(e) The three graphs show the histogram, probability density function and the empirical cumulative distribution function for the training set data of feature05, respectively. The bar chart in (f) gives the importance for each feature in decision making. (g) shows the receiver–operator curve for samples of class “1” and the calculated area-under-the-curve. The confusion matrix in (h) shows the four possible classification outcomes for the example binary classifier and how many samples fall into each. The classification outcomes are used to calculate several quality metrics (classification accuracy and error, sensitivity, specificity, false-positive rate, precision, F1 score, ROC AUC) as can be seen in the radar plot in (i).
Figure 6. Continued.

for a feature given in dark blue showing that a feature is perfectly positively correlated with itself, i.e. ‘1’ or 100%. In Figure 6(c–e), feature05 is plotted as a histogram (c)), a probability density function (PDF; d) and an observed and theoretical (empirical) cumulative density function ((E)CDF; e) with the mean (red vertical line) and median (green vertical line) calculated for the feature marked for the latter two. The histogram and probability density function show a clear skew to the left (lower x-values) and the corresponding cumulative density function does not follow the expected, theoretical sigmoidal shape. This feature would benefit from some form of feature engineering, e.g. normalization, to achieve a Gaussian distribution of the data. An example from the crystallographic field for the effect of skew in the data is histogram matching to identify electron density maps representative for a protein with well-defined atomic coordinates and a connected backbone [97].

**Find attributes/features/variables in the dataset**

Attributes/features/variables of the data define the dimensions of a dataset and can be placed into one of two groups, numerical or categorical. Numerical data usually hold quantitative information about a dataset whereas categorical data provide qualitative information. Attributes/features/variables that are given as categorical data cannot be analysed through summary statistics in particular if there are many individual categories available and other statistical methods may not produce optimal results. Combining several categories into a group is often used to address the problem. This is
particularly important when considering class distribution and balance in a classification problem.

**Conduct univariate data analysis to identify data distributions**

Calculating simple summary statistics for each attribute/feature/variable and plotting them individually in varying ways is the first part of this analysis. The focus is on identifying the types of data distributions found for the different attributes/features/variables, their centrality (mean, median, mode) and their dispersion (range, variance, standard deviation, skew, kurtosis). Table 1 contains the summary statistics for the data used to draw the plots in Figure 6. For feature05 for example, the arithmetic mean (sum over all values for this feature divided by the number of samples) is 35.78 and its median (the middle value for the ranked samples) of 28.67. The mode for feature05 is 1, meaning that there is one value in the distribution that is found most often. The range of the data of feature05 is defined by its minimal and maximal values which are 14.19 and 100.00 respectively. A standard deviation can be useful to calculate if the data are normally distributed to determine how much the samples in the distribution on average deviate from the mean. For feature05, we find 18.42, although this is meaningless as we can see from the density distribution plot in Figure 6(d) that the samples for this feature are not normally distributed and some normalization should be applied. The inter-quartile range can be used to assess the spread or dispersion of samples if they are not normally distributed. Therefore, in addition to the median, the 25% and 75% quantile are calculated, for which we find 22.51 and 43.54. The difference between them is the inter-quartile range, here 21.03. A small value for the inter-quartile range means a small dispersion of the samples. The skew describes whether the average or mean can be found in the centre of the plotted sample distribution. For feature05, a skew of 1.49 was calculated, which means that the sample distribution is skewed to the left with a long trail towards the right as can indeed be seen in Figure 6(d). A negative value indicates a shift to the right with a trail to the left. For normally distributed data no skew is detectable and the value should be 0. The kurtosis describes the pointiness of the sample distribution or how well the samples cluster around the centre. For feature05, the kurtosis is 1.82 and Figure 6(d) has a fairly pointy appearance. A negative value would indicate a rather flat sample distribution with a broad spread and probably a large standard deviation. The most commonly found data distributions are normal (Gaussian), Poisson and binomial distributions. For many machine learning applications, normally distributed data are preferred and normalization should be applied to achieve this.

**Use bivariate/multivariate data analysis to identify relationships and interactions**

Pairwise analysis and multivariate statistics are used to identify interactions and dependencies between the different attributes/features/variables. Visual inspection of the data in pairwise scatterplots or correlation matrices after calculating Pearson’s correlation coefficient is common. The interpretation of results from multivariate analysis of several attributes/features/variables can be a challenge and often requires domain knowledge of the data. It often requires some time and computational effort and the use of factor analysis techniques such as clustering or dimensionality reduction (see above).
Table 1. Summary statistics from data exploration step for example data.

| Feature   | Minimum | Maximum | Sum      | Mean | Median | Standard deviation | Variance |
|-----------|---------|---------|----------|------|--------|--------------------|----------|
| feature01 | 11.40   | 100.00  | 96491.17 | 94.60| 99.11  | 10.83              | 117.19   |
| feature02 | 19.92   | 150.20  | 48325.97 | 47.38| 43.46  | 14.75              | 217.69   |
| feature03 | −265.75 | 135.72  | 27589.24 | 27.05| 23.75  | 17.71              | 313.65   |
| feature04 | 1.44    | 8859.80 | 475782.90| 466.45|302.10  | 616.14             | 379628.70|
| feature05 | 14.19   |100.00   | 36495.72 | 35.78|28.67   | 18.42              | 339.43   |
| feature06 | 26879.0 | 22807940.0 | 800712200.0 | 785011.93 | 555324.50 | 1140006.00 | 1299614000000.00 |
| feature07 | 0.30    | 0.66    | 519.86   | 0.51 |0.52    | 0.07               | 0.01     |
| feature08 | 9821.00 | 500916.00 | 48422080.00 | 47472.62 | 36806.00 | 40210.58 | 16168910000.00 |

| 25% quantile | 50% quantile | 75% quantile | skew | kurtosis | mode |
|--------------|---------------|---------------|------|----------|------|
| feature01    | 95.40         | 99.11         | −3.27| 12.62    | 1    |
| feature02    | 37.74         | 43.46         | 53.04| 1.87     | 5.92 | 1020 |
| feature03    | 16.32         | 23.75         | 35.02| −3.60    | 74.00| 9    |
| feature04    | 98.60         | 302.10        | 545.80| 4.59     | 40.27| 7    |
| feature05    | 22.51         | 28.67         | 43.54| 1.49     | 1.82 | 1    |
| feature06    | 325733.00     | 555324.50     | 928986.00| 10.49    | 166.02| 1    |
| feature07    | 0.46          | 0.52          | 0.56 |−0.29     |−0.37|3     |
| feature08    | 23812.00      | 36806.00      | 55337.00| 4.20     | 32.41|3     |

**Detect missing values**

For rows containing null values, if a large enough dataset is available, such samples are usually removed. If sample size is limited, then data can be imputed by, for example, replacing the missing value with the mean or median calculated for the corresponding attribute/feature/variable. Missing data can also be visualized in plots, e.g. if certain areas reveal gaps. In crystallography, this can be compared to filling missing reflections that could not be measured in an experiment or to replace those which have large measurement errors to improve the chances of successful phasing [98].

**Detect outliers**

Any form of outliers needs to be removed by deleting the corresponding sample. It is also possible to set minimum/maximum boundaries for the attributes/features/variables and exclude certain ranges of data if they are not reliable. Visual assessment by using plots very quickly identifies outliers. Calculating Inter-Quartile Range [99] helps to identify outliers within a single feature. Correlation coefficients [100] and factor analysis [95] can be used to identify outliers after bivariate and multivariate analysis respectively.

**Feature engineering**

The main purpose of this step is feature creation and transformation. A very good introduction can be found in [101]. The features as found in the raw data may not be descriptive enough to identify a pattern and some new features may have to be created. Often, features will be combined, for example, in a linear fashion or into polynomials. Such combinations dramatically reduce the complexity of data analysis. Other feature transformations may be developed making use of domain-specific knowledge and help to turn non-linear relationships between features into linear ones. Standardization is used to improve the understanding of the data whereas normalization turns skewed data distributions into
symmetric ones. If a variable is continuous, then in some cases binning and discretization may be necessary. This is similar to defining a resolution range for crystallographic data and then dividing the range into bins so that each bin contains approximately the same number of reflections. Features may be excluded through systematic trial-and-error approaches or data dimensions may be reduced by looking at dimensionality reduction procedures mentioned above.

These steps may be repeated several times to get a better insight and may require rescaling in between the different analyses.

**Performance measures and producing robust results**

Working with ML and AI algorithms is a purely data-driven approach. No initial assumptions, exclusions or biases about the systems one aims to develop should be made.

**Performance measures**

If a supervised classification problem is addressed with a machine learning algorithm, then a confusion matrix is used for performance assessment (see Figure 6(g–i)). In a simple binary case, all correctly identified positive and negative samples are recorded, true positive (TP) and true negative (TN), respectively. Additionally, samples for which the prediction and the known ground-truth disagree are counted as false-positive (FP) and false-negative (FN). All positive samples are given as \( P = TP + FN \) and all negative samples as \( N = TN + FP \). Using the counts for the different binary classification outcomes, the following metrics can be calculated: classification accuracy and error, sensitivity, specificity, false-positive rate, precision and F1 score (a metric for test accuracy based on the harmonic measure of precision and sensitivity/recall).

The ‘Classification Accuracy’ \( (ACC) \) is defined as

\[
Classification Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (1)
\]

and gives an overall performance measure of the predictor for both classes in the binary classification problem.

The ‘Classification error’ gives details about misclassification events and can be calculated as

\[
Classification error = \frac{FP + FN}{TP + TN + FP + FN} \quad (2)
\]

or as \( 1 - ACC \). A classification error of 5% is often used as a benchmark as this is the typically observed human classification performance \([102]\).

The ‘Sensitivity’, recall or true-positive rate \( (TPR) \) is a measure to judge how well a predictor does in correctly identifying samples of the positive class out of all positive cases and is defined as

\[
Sensitivity or TPR = \frac{TP}{P} = \frac{TP}{TP + FN} \quad (3)
\]

The ‘Specificity’ or true-negative rate \( (TNR) \) looks at the predictor’s performance in correctly identifying all negative samples out of the total number of negative class samples.
and is calculated below

\[
\text{Specificity or } TNR = \frac{TN}{N} = \frac{TN}{TN + FP} \tag{4}
\]

The ‘False-positive rate’ (FPR) gives a measure for how many samples of the negative class have been predicted to wrongly belong to the positive class

\[
\text{False - positive rate or } FPR = \frac{FP}{N} = \frac{FP}{TN + FP} \tag{5}
\]

Specificity can also be calculated as 1 – FPR and the false-positive rate is related to this as 1 – TNR.

The ‘Precision’ or positive predictive value (PPV) looks at how many of the samples belonging to the positive class have been correctly identified as such out of all samples assigned to the positive class

\[
\text{Precision or } PPV = \frac{TP}{TP + FP} \tag{6}
\]

The ‘F1 score’ is a trade-off between ‘Precision’ and ‘Sensitivity’. It looks at how well a predictor does in identifying samples of the positive class while also considering the misclassification outcomes of false-positives and false-negatives. It is calculated as follows:

\[
F1 \text{ score} = \frac{2}{\frac{1}{\text{precision}} + \frac{1}{\text{sensitivity}}} = \frac{TP}{TP + \frac{FN=FP}{2}} \tag{7}
\]

A good binary classifier will score close to 1 or 100% for the ‘Classification accuracy’ and the ‘F1 score’, although the latter can be significantly lower depending on whether a classifier is wanted, to have high ‘Precision’ or ‘Sensitivity’. ‘Classification error’ and ‘FPR’ are expected to be close to zero or 0%, whereas ‘Sensitivity’, ‘Specificity’, and ‘Precision’ should all score close to 1 or 100%. In general, a model cannot achieve perfect predictions unless it has learned the data.

Additionally, the area-under-a-curve of a receiver-operating characteristic (ROC) can be calculated. In an ROC plot, the false-positive rate, FPR, is plotted on the horizontal axes whereas the true-positive rate, TPR, is plotted on the vertical axes. In the case of a well-performing binary classifier, the curve will exhibit a peak near the top left corner where FPR is minimal and TPR at its maximum. For the area-under-the-curve (AUC) determined for an ROC curve, an AUC value close to 1 or 100% should be found if the system is performing well [103].

Figure 6(h) and (i) give a confusion matrix and radar plot of performance metrics for a binary classification predictor of a decision tree with an AdaBoost algorithm. Such a system is called an ensemble predictor as the combined decisions of hundreds or thousands of trees are evaluated. Classification accuracy and error are global metrics describing the overall performance of a classifier. Here, we find 96% for the classification accuracy and 4% for the error when challenged with the test set, data that was kept separate and has not been used in training. On the whole, our system performs well. For sensitivity and specificity, we find 94% and 97%, respectively, which means that the predictor performs almost equally well in correctly classifying samples for both, the positive and the negative class. This is further
Figure 7. Sample distribution of a bootstrap calculation to determine the confidence interval of a predictor.

supported by a small false-positive rate of 3% and a high score for precision of 94%. An ROC curve is plotted in Figure 6(g) and the area-under-the-curve has been calculated to be 99% for this example system with a peak close to the top left corner. As the system used here represents an ensemble of decision makers, the importance of each feature for each ensemble member can be accessed as was plotted in Figure 6(f). Feature05 seems to be of high importance across the different decision makers in the ensemble.

Above we looked at measures to judge the performance of a single algorithm in a classification problem. As was mentioned previously, it is impossible to say at the beginning which algorithm will be best suited to solve one’s problem. A way of comparing the performance between different algorithms is using a statistical significance test like Student’s $t$-test [104] or calculating a confidence interval using a bootstrap method [105] and applying a pairwise comparison between the different algorithms. Bootstrapping is a resampling method and uses random sampling with replacement to find the sample distribution in the available data. The 95% confidence interval for the example system used here is in the probability range from 51% to 65% and the corresponding sample distribution for the underlying bootstrap calculation is given in Figure 7. For our predictor, there is a 95% chance that a sample lies in a range from 51% to 65% for its prediction probability. For a pairwise comparison, the different algorithms have to be run on the same set of data, i.e. the same split into training and testing data, and should be combined with k-fold cross-validation (see below).

For deep learning applications, a system’s performance is not only judged by classification outcomes alone (see above) but also by looking at the behaviour of the learning curve over time. In fact, all deep learning models, regardless of whether they have been trained for
Figure 8. For deep learning applications, the training and performance of a predictor are often assessed using learning curves, the loss and model accuracy in particular. The left column (plots a–d) shows the loss over time when training a convolutional neural network and the right column (plots e–h) shows the model accuracy for the same model.
a classification or a regression problem, make their final prediction by means of regression by applying a minimization function, usually stochastic gradient descent. The goal of this minimization is to reduce the cost or penalty in the system by correctly identifying samples and returning the information of the outcome through backpropagation so weights can be adjusted for the next training cycle. After each training cycle, this can be a batch of data, if the total input is too large to be held in memory, and/or at the end of a training epoch, the system is evaluated by using a validation set and the loss is recorded and plotted for the training and validation set. Monitoring the loss function by plotting a learning curve is therefore a good way to see whether the system performs according to expectation and if there is any behaviour that is of concern, e.g. over- or underfitting. For a system where training is working well, the model will gain experience over time and the learning curve will converge towards a maximum with the loss approaching a minimum. Figure 8 shows some learning curves and how problems manifest themselves. Plots (a) and (e) give the behaviour for a good system where the curves for the training and testing set follow the same pattern overall and where there are few extreme fluctuations. A system where the model performs better on the training data than on the testing data is overfitting and has learned the data. The loss for the training data is lower (b) and the model accuracy higher (f) when comparing to the testing data if the system is overfitting. On the other hand, as one can see from plots (c) and (g), if a model is underfitting, perhaps because it has not run long enough to reach convergence, then for both, the training and the testing set, the loss is high and the accuracy low. If the split into training and testing set is set such that the number of samples for training is small compared to the number of samples for testing, then both curves will follow a similar trend but with a large gap between them as in (d) and (h). As with the performance measures introduced above, a perfect system that has learned all the data and has optimized all the internal parameters for this data will make no mistakes and has a loss of ‘0’. Such a system is expected to fail to generalize when challenged with new samples unless they represent a perfect match to the learned data. For classification problems, a second learning curve can be plotted which monitors the classification accuracy (see above) as a second metric. An additional testing set is often used as a final assessment after all the training as finished and above performance metrics are calculated.

Robustness

To achieve robust results, the data used for training needs to be presented to an algorithm in multiple ways to ensure that there is consistency between the individual results. First, one needs to consider the way data have been split into a training and testing set. Commonly used splits are 80%, 75%, 70% of the samples for the training set and 20%, 25%, 30% for the testing set. These splits are generally applicable but may need adjusting based on the problem at hand. It is also important to consider the distribution of samples. Generally, any predictive tool works on the assumption that the data have an equal distribution for the different outcomes, i.e. if classes are to be predicted then all classes have to be present equally or for regression it is expected that samples cover the data range smoothly with few extreme outliers.

If balanced data distributions cannot be achieved, and they very rarely can, then the imbalance needs to be addressed when splitting the data. Commonly, one applies stratification, a method that ensures that the overall data distribution is retained in the training and
testing data. Alternatively, minority samples can be ‘boosted’ in a process called ‘oversampling’ by creating simulated or synthetic new instances which consider the actual nature of the data as a starting point. A summary and assessment of various oversampling techniques is given in Batista et al. [106]. On the other hand, one can also reduce the number of samples in the dominant class/es through ‘downsampling’ by randomly eliminating samples. If too many samples are removed, this will however affect the stability of a predictive system.

K-fold cross-validation [107,108] is another way to increase the stability of a system. Several cross-validation folds are created by taking a subset of data from the larger training set. The system is set up such that each sample can only be used once in a cross-validation fold but can appear multiple times in the training data depending on how many folds have been chosen. In this procedure, it must also be insured to apply stratification to have persistent data representation in the different folds. If there are large differences in performance between the different cross-validation folds, then the system is considered unstable and the algorithm and/or parameters need to be evaluated and adjusted.

Unlike machine learning where model robustness is usually achieved through cross-validation, in deep learning randomness is the preferred option. As neural networks in deep learning are stochastic, random initial weights at the beginning of a training cycle and shuffling the data before each training epoch are the means by which robustness is ensured for the model. The results will differ for each cycle but not to the extent that they are contradictory and over time the system will converge to a minimal value. Validation done with a hold-out set of data will produce similar results each time. Splitting the data into training and testing set to account for model variance still applies, as well as ensuring that both sets are representative in terms of class distribution when compared to all the data. Internally a deep learning model does have an additional layer of randomness through shuffling and initial weights. A random seed can be defined to give some level of control and robustness if reproducibility is desired. To show robustness when evaluating a deep learning model for realistic applications, training and evaluation should be repeated multiple times and then standard deviation, standard error and a confidence interval be determined using all the model performances.

The easiest way to explore a large number of options in a data-driven way is to use automation. Here, a pipeline can be created that tries all algorithms one wants to evaluate and changes parameters either systematically through a grid search or by randomly selecting values from a defined range [109].

**Examples of machine learning in the field of biological crystallography**

Here, we will look at applications of algorithms from the fields of machine learning and artificial intelligence in six areas of protein crystallography: serial crystallography, software for model building and refinement, decision making in automated data analysis at synchrotron facilities, protein crystallizability, crystallization outcome classification and structure prediction. A summary of the different applications is given in Table 2.

**Serial crystallography**

In serial crystallography a set of diffraction images is produced by irradiating hundreds or thousands of small protein crystals, sometimes less than a micron in size, with a
**Table 2. Summary table of the discussed machine learning and AI applications.**

| Field of application | Model/software package | Architecture | Public data | Citation |
|----------------------|------------------------|--------------|-------------|----------|
| Diffraction image classification | DeepFreak | ResNet-50 | DiffraNet | Souza et al. [86] |
| Diffraction image classification | – | AlexNet | CXIDB | Ke et al. [87] |
| 3D model building | Buccaneer; Coot | Multi-layer perceptron | PDB | Bond et al. [83] |
| 3D model building | ARP/wARP | Unidirected nearest-neighbour graph; Support vector machine | PDB | Chojnowski et al. [65] |
| 3D model building | ARP/wARP | Principal component analysis | PDB | Pereira and Lamzin [46] |
| 3D model building | ARP/wARP | Linear discriminant analysis | PDB | Hattne and Lamzin [47] |
| Decision making in automation | METRIX | Decision tree with AdaBoost | PDB | Vollmar et al. [76] |
| Protein crystallizability | SECRET | Support vector machine; Naïve Bayes classifier | PDB; TargetDB | Smialowski et al. [55] |
| Protein crystallizability | XtalPred | Logarithmic opinion pool | PDB; TargetDB | Slabinski et al. [110] |
| Protein crystallizability | XtalPred-RF | Random forest | PSI TargetTrack database (http://sbkb.org/index.jsp) | Jahandideh et al. [74] |
| Protein crystallizability | PPCpred | Support vector machine | PepcDB | Miziantsy and Kurgan [62] |
| Protein crystallizability | PredPPCrys | Support vector machine | PepcDB | Wang et al. [63] |
| Protein crystallizability | DHS-Crystallize | Convolutional neural network; XGBoost | Wang et al. [63] | Alavi and Ascher [75] |
| Protein crystallizability | DeepCrystal | Deep convolutional neural network | Wang et al. [63] | Elbasir et al. [84] |
| Crystallization outcome classification | MARCO | Deep convolutional neural network | https://marco.ccr.buffalo.edu [20] | Bruno et al. [20] |
| Crystallization outcome classification | – | Linear discriminant analysis | – | Cumbaa et al. [48]; Cumbaa et al. [49]; Saitoh et al. [50] |
| Crystallization outcome classification | – | Linear discriminant analysis | – | |
| Crystallization outcome classification | – | Decision tree, Random forest | – | Bern et al. [72] |
| Crystallization outcome classification | – | Decision tree, Random forest | – | Liu et al. [73] |
| Crystallization outcome classification | ALICE | Support vector machine; Linear discriminant analysis | – | Buchala and Wilson [51] |
| Crystallization outcome classification | – | Support vector machine | – | Pan et al. [64] |

(continued)
high-intensity beam, often an X-ray free-electron laser. In the process of the experiment, in many cases, the protein crystal gets destroyed or is severely damaged allowing only a single or few image/s per crystal to be recorded. For data analysis to calculate a protein’s electron density to be able to build a model, only the images providing useful data should be combined. In particular, images that exhibit no or very noisy diffraction and those that may show pathologies that are currently difficult to correctly address, need to be excluded. Identifying the most suitable set of images for further analysis is currently an entirely manual and laborious process. Due to the very large data rates and data volumes generated by serial crystallography, there has been a demand for methods to rapidly classify and triage measured images to, first, determine which data to use for further analysis and, second, to determine which data must be stored and which can be discarded.

Souza et al. [86] investigated the use of machine learning applications to identify and classify serial crystallography images so that they can be combined for further downstream analysis. The data used to train several machine learning algorithms were a combination of real and synthetic diffraction images, DiffraNet. In total, five classes were predicted for the synthetic images, all of them representing possible outcomes for a diffraction experiment. The real images were split into two classes, contain diffraction or do not contain diffraction. Of the three algorithms tested, random forest, support vector machines and a convolutional neural network based on a ResNet-50 architecture, the latter performed best on both sets of images as judged by classification accuracy. The network was named DeepFreak. The prediction accuracy to identify images belonging to the five different classes was higher for the synthetic images than for the real images with 98.45%, 97.66% and 98.5%, respectively, for the three different algorithms tried. For the real images, the three different approaches achieved an accuracy of 86.81%, 91.1% and 94.51%, respectively. The lower performance on the real data ought to be expected, as it is difficult to simulate all possible sources of noise and error in an experiment and therefore synthetic images will be easier to learn and to predict with the predictor achieving higher scores.

Another convolutional neural network-based approach was developed by Ke et al. [87]. The algorithm used the architecture of AlexNet [113]. The data used for training are real image data collected at the Linac Coherent Light Source (LCLS). As in Souza et al. [86] several classes were predicted based on the potential outcome for serial crystallography data collection, in this application three in total. The labels for these classes were assigned in two ways, by a human expert crystallographer and an automated system using a spot-finding algorithm with a threshold. Five datasets representative of commonly acquired data have

### Table 2. Continued.

| Field of application | Model/software package | Architecture | Public data | Citation |
|----------------------|------------------------|--------------|-------------|----------|
| Proteins modelling   | RaptorX                | ResNet       | PDB         | Wang et al. [88] |
| Proteins modelling   | AlphaFold              | ResNet       | PDB         | Senioret al. [88]; Senioret al. [89]; Senioret al. [90] |
| Proteins modelling   | AlphaFold2             | Attention based network | PDB; various sequence databases | Jumper et al. [111] |
| Proteins modelling   | trRosetta              | ResNet       | PDB         | Yang et al. [91] |
| Proteins modelling   | RoseTTAFold            | 3-track network | --         | Baek et al. [112] |
been used for training. The images have been made publicly available in the Coherent X-ray Imaging Data Bank (CXIDB; [114]). Between 70% and 98% of the images across the different datasets were given the correct label by the CNN, whereby the human annotation served as the ground truth. These success rates were highly specific to the experimental setup used when collecting the images. The authors already discussed this, as they trained a CNN on a set of images acquired on one type of detector but made predictions on images collected on a different detector type which results in lower performance. However, this was improved when applying some image pre-processing techniques such as local contrast normalization [115] and data augmentation through random cropping. The more interesting aspect here is, that the authors also attempted automated spot finding and compare the results to the automated step in the diffraction data integration package DIALS [116], which can be applied to diffraction data collected on a large number of sources. Although, CNN-based automated spot finding shows generally lower performance, up to 86% of the images showing diffraction can be identified and nearly all images without any diffraction. It would be interesting to see how this classifier can generalize when given images collected at other XFEL facilities.

Usage in model building and refinement software packages

Here, we are looking at multiple ways machine learning and artificial intelligence have been used in programs employed for building protein structures by placing atoms into experimentally derived electron density maps.

We first look at Bond et al. [83]. They train two neural networks (both a multi-layer perceptron), one to look at the protein backbone or main chain and one for the side chains. The target is to predict a new correctness score for how well the atoms have been placed into the electron density map during model building.

The input layer consisted of one node for each feature (different for main chain and side chain features), followed by a single hidden layer with ten neurons and an output layer predicting a single value, the correctness. The data used for training were selected from the PDB [6]. The features used are common crystallographic model quality metrics, here, calculated using tools implemented within the model building and visualization program Coot [117,118]. All features were transformed to a mean of 0 and unit variance. The mean and standard error in the coefficient of determination (COD) for the test set was used to assess the performance of 100 training repeats each time with a randomly chosen starting seed. As a score between 0 and 1 was the prediction outcome, the machine learning problem was that of regression. Additionally, the authors applied a threshold for the score of \( \geq 0.5 \) and turned the regression into a binary classification problem.

Using performance metrics for classification to assess the predictor showed that the results for the main chain predictions were more reliable than for the side chain with 92.3% and 87.6% of the atoms being given the correct label, respectively. The correct label, in this case, meant that an atom had been found in its proper position in the 3D structure and the predictor was able to identify this, or that an atom was flagged to require attention because it had been placed in the wrong position during automated model building. True-positive samples were those, where the atoms were built well into the electron density, and true-negative samples were those requiring manual adjustment. The fit was given by the correlation between the model and the fit into the electron density, Z-scores for the mean,
best and difference density, Z-scores for the B factors, overlaps between atoms and the resolution. For the main chain predictor additionally, Ramachandran score and the twist in the peptide plane and flip of its nitrogen and oxygen atoms were recorded. The predictor for the side chains additionally used a rotamer score. Both predictors showed high false-positive rates, 23% for the main chain predictions and 21% for the side chain predictions, which means that both perform less well in identifying atoms that may require some attention by the experimentalist. The most likely reason for this is that the training data had very few examples of wrongly placed atoms and perhaps the imbalance was not treated appropriately in the training process. Nevertheless, overall, both predictors helped to improve automated structure building which in turn will reduce the time a crystallographer may need to manually adjust a model.

The predicted correctness scores for each atom can be visualized in Coot and an automated pruning function for chains, residues and side chains with low results has been implemented in the automated model building pipeline Buccaneer [119,120]. In particular, using the correctness scores has resulted in significant improvements for high-resolution structures. This method is a first step towards improving automated model building using artificial intelligence compared to the currently available programs.

Another crystallographic software suite making use of machine learning techniques at different stages of model building is AR/wARP. One very recent development is the use of machine learning to assign fragments of protein backbone [65]. Those fragments are usually the result of automated model building and finding the correct sequence register is particularly difficult for low-resolution data.

A database of protein fragments extracted from the PDB was used to train two machine learning applications. As with Bond et al. [83], the two applications are addressing main-chain and side-chain atoms separately. Good quality high to medium resolution structures were used for training. The structures used for training were broken into fragments of differing backbone length and grouped by similarity using superpositioning. Testing was done for two separate groups, one covering medium to low-resolution structures and one for low to very low resolution. The number of samples in each group was balanced by reducing those in the first test set through random selection from equally sized bins. After attempting automated model building without a supplied sequence only the top 50% of solutions, after comparing with the deposited ground-truth structures, was kept for performance analysis of a machine learning-based sequence classification tool.

If the deposited structure, the ground-truth and the built model agreed with each other based on main chain and side chain specific criteria, then a $\alpha$ or side chain was labelled ‘correct’. The loop regions were built using location probabilities. Side chain descriptors were created using the top 500 rotamers library [121] and the different conformations aligned on the peptide plane with the built residues. A Cartesian grid was placed on $\alpha$ after alignment with a spacing of 1Å between the grid points. The grid points that matched the electron density region at a given threshold were used to construct an undirected nearest-neighbour graph. The dimensions reached through connectivity within the graph was then compared to the density volume at varying levels of root-mean squared deviations.

Including treatments for variation in the data due to experimental and data analysis effects, a final side chain descriptor was a vector of 25 elements. Factors influencing the side chain descriptor were solvent content, data resolution, Wilson B factor and phase quality as
they directly alter the shape of the electron density. It was found that for some side chains, e.g., threonine and valine, the side chain descriptors were very similar as they filled a similar volume and shape of electron density, but they had different mobility properties. Determining the side chain descriptor was a key data preparation step for the successful training of a support vector machine as a one-versus-all classifier. The amino acid type for a residue within a main-chain fragment was then determined using a separate system in which a separate classifier for each amino acid had been produced. It estimated the probability for each of the 20 different amino acids independent of the sequence information given for the protein. The 20 different classifiers used a soft-margin and a radial basis function (RBF) kernel. The RBF kernel uses the squared Euclidean distance between two feature vectors to describe similarity. The closer the two feature vectors are to each other in the feature space the greater is the squared Euclidean distance and the more similar are the two samples described by the feature vectors. A sigmoidal function was used for calibration of all the 20 classifiers. Running the set of 20 classifiers on the main chain fragment gave probability estimates for each type of residue in the fragment. For the entire fragment itself, this was then recorded in a statistical scoring matrix. The matrix was then used to align the fragment with the target sequence and approximate probabilities for each residue in the alignment were determined. Alignments reaching a confidence level of 99.99% were scored as accepted. Comparison was then made to an alignment between the fragment and a randomly chosen sequence following the same protocol. All accepted alignments were then assigned to the target sequence using a directed graph, where the nodes represented the aligned fragments and the connections between them were directed if they could be assigned to the same chain. The lengths of the edges represent real-space distances between Cα atoms at the ends of the fragments. Probabilities were calculated for the different possible paths through the graph when connecting the fragments with the highest probability being the most likely path and the most likely sequence assignment. The method worked well for medium to low-resolution test structures with somewhat lower performance for very low-resolution models down to 4Å. The predictors performance strongly correlated with side chain mobility, where the latter affected the local quality of the electron density map which in turn influenced the undirected graph used to identify amino acid types. Short, well-ordered side chains worked well with classification accuracies of 96%, 93% and 72% for glycine, alanine and proline respectively. Large, aromatic ones like tyrosine and tryptophane were identified with accuracies of 67% and 60% respectively. Residues exposed to solvents on the protein surface or in solvent channels were the least likely to be correctly identified with accuracies of 5% for lysine, 9% for asparagine and 18% for histidine. For both test sets, the very low-resolution set, in particular, the number of models with low side chain completeness was reduced and the number of models with a large number of side chains correctly built was increased. In combination with the explored loop building algorithm, the total chain length increased substantially and, overall, the model building performance of ARP/wARP was greatly improved. Two novel examples given in the paper taken from the package’s webservice platform, exhibited a striking improvement in model quality for the new methods. For one example given in the publication, this meant the refinement R/R_free factors improved from 31/36% to 25/29% and for another the change was from 37/45% to 24/29%. From a machine learning point of view, the implementation followed commonly used good practice, by trying to balance the sample distributions between different resolution ranges explored and by limiting the number of support vectors.
to avoid overfitting. Confidence intervals were provided to demonstrate the robustness of the system.

Another example implemented in ARP/wARP is a validation tool for protein main chain conformation by Pereira and Lamzin [46]. The data for developing the application was again based on structures in the PDB. Dipeptide blocks were extracted from good quality released structures and subjected to some filtering to ensure extracted blocks were reliable. Pairwise sequence identity was less than 50% to ensure non-redundancy. For each dipeptide, a Euclidean distance-squared matrix was calculated for a set of distances used to describe the peptide plane in a protein main chain. Principal component analysis was then used to create a new Euclidean orthogonal 3D-space, comprised of the three decorrelated principal components. This space was further divided to account for chirality in molecules. This newly created space and its subdivision can be used to assign probabilities to the backbone geometry. To do so, negative examples were created by randomly placing atoms into a defined volume and distance matrices and eigenvalues were calculated as before. To define the boundaries for favoured, allowed, generously allowed and disallowed scores, similar to what is found in a Ramachandran plot [122], a cumulative density distribution was calculated for all grid points. Z-scores were calculated for mean, variance, skew and kurtosis for a set of randomly selected structures, assuming that the scores found for Cα atoms were following a Gaussian distribution. These Z-scores were then further analysed with PCA to decorrelate the different moments and combine the scores into a new scoring function. This new scoring function was calculated separately for each of the boundaries defined above. This method increased the understanding of backbone geometry further than the commonly used Ramachandran plot. And indeed, the authors found that angles τ, ϕ and ψ commonly used to judge the geometric quality of a residue in the Ramachandran plot were not directly correlated to their newly defined score. Instead, it was observed that the first principal component described the extension of the dipeptide, the second principal component gave the twist between the two adjacent peptide planes and the third principal component reported bending in the dipeptide. Applying the knowledge gained from the PCA analysis of the distances in dipeptides within a protein structure identified areas in the main chain with distorted geometry, which either required adjustment or careful assessment whether this effect was of catalytic importance.

The last example for an application within ARP/wARP is described in Hattne and Lamzin [47]. Here, the authors developed a pattern-recognition method to identify planar objects in proteins (main chain peptide; aromatic side chains such as histidine, tryptophan, tyrosine and phenylalanine) or the base moiety in DNA/RNA structures. Aromatic side chain identification was of particular interest as it served as an anchor point for sequence docking from which automated side chain building along the backbone commenced. Features were generated locally from electron density maps and then classified using linear discriminant analysis by optimizing the contrast between planar and non-planar objects. It relied on the fact that a particular arrangement of atoms was surrounded by a specific electron density shape. To extract these local features, spherical regions in the normalized density were sampled and combined into a single feature vector. The size of the sphere used, determined the type of information found in the feature vector. Too large a volume created overlaps between adjacent planes and too small a volume missed information. A single size of 3Å to extract all features at once was determined empirically which resulted
in a trade-off between the different classes and caused all of them to have a certain level of false-positives. The authors considered trying a specific sphere size for each class in their future plans. Agreements between feature vectors in the training set and those extracted from new samples gave an idea about the planarity in the local part of the electron density encoded in the vector. After applying transformations and normalization the resulting feature vectors were classified using a linear discriminant analysis [44, 45]. The four classes to identify were small single-ring objects (histidine), large single-ring objects (single-ring nucleotide bases, phenylalanine, tyrosine), double-ring objects (double-ring nucleotide bases, tryptophan) and noise. Overall, the method identified the majority of double-ring objects, between 42% and 100%, and large single-ring objects, between 50% and 93%, for the five case studies presented. For small, single-ring objects between 42% and 70% of the cases were found. As there was a resolution dependence, the authors found a reduction in performance with a drop in resolution. It was also noted that a single structure was used to extract the feature vectors for training which reduced the variance in training samples. To improve the method, a larger, more diverse set of feature vectors needed to be created looking at structures of differing quality and overall resolution or even better local resolution. Wilson B factors, or more likely the average temperature factor found for each ring object, should be included as the method looked at local electron density features and the same plane element had varying electron density in different parts of the protein or between non-crystallographic symmetry copies. The authors noted that in general there were only few planar objects in a protein structure which they saw as a low signal-to-noise ratio and it being challenging to identify such systems. In fact, for two of the cases presented, the noise level was high enough to cause the algorithm to find twice as many false-positive samples than true-positives. Also, each planar object extracted from the selected structure was regarded as a training sample which in turn meant that the different classes that needed to be identified were not present in equal numbers. As a result, the system published was biased towards identifying double-ring objects. Nevertheless, it was implemented for automated model building in ARP/wARP and already provides good starting points for a crystallographer to then manually complete the initial model.

**Decision making in automated data analysis pipelines**

Large-scale user facilities, like synchrotrons, produce huge volumes of experimental data every year. One of the key challenges in handling such data volumes is their analysis, as the computational resources are finite and typically preclude making brute-force attempts at structure determination for each data set that is measured. As discussed earlier, the trained crystallographer is usually able to make an educated assessment as to the quality of data and the likelihood of it leading to successful structure determination or that it would answer the scientific question being asked. No single quality metric of crystallographic data can be relied upon to support this assessment. On the contrary, a crystallographer makes an assessment based on several indicators of quality. Ultimately, the crystallographer wants to avoid wasting time on analysing poor quality data that will yield no new information or results; better to re-measure the data or add more data.

The authors of this review explored in Vollmar et al. [76] machine learning applications to triage and assess experimental data and their metadata at early stages of data analysis to make more intelligent use of the computational resources for down-stream analysis.
The authors focused on the case of experimental phasing using single/multiple anomalous dispersion (S/MAD). They explored commonly used data quality metrics, which are available from the integration of raw experimental diffraction data and subsequent scaling and merging steps. The raw diffraction images used to produce the features for training were publicly available and the original structure factors and atomic coordinates were deposited in the PDB. A wide range of structure size, resolution, type of protein and hardware the data were collected on were covered. The diffraction data were integrated, and phases were calculated using commonly available crystallographic software while largely keeping default settings. Metadata in the form of common crystallographic data quality metrics that were to be used as features, were extracted and stored in a small database for ease of management. A set of statistics/features was identified from a large number of extracted metadata using exploratory data analysis and initial machine learning trials (SVM, decision trees, ensemble predictors using bagging and boosting). The features were chosen based on their importance in decision making as identified by EDA and the tried algorithms themselves, i.e. data-driven rather than by domain knowledge and exclusion or prioritization. In total, six features were identified; $CC_{\text{anom}}$ (correlation coefficient between Bijvoet differences), $\Delta I/\sigma I$ (fractional anomalous intensity difference), $m_{\text{anom}}$ (mid-slope of the anomalous normal probability), $\Delta F/F$ (fractional anomalous difference based on structure factor magnitudes), $f''_{\text{theor}}$ (theoretically determined value of $f''$, imaginary part of anomalous scattering coefficient for the anomalous scatterer) and $d_{\text{max}}$, (the low-resolution limit of the data). All but the last descriptor were expected to be important for experimental phasing as they are recognized indicators of the presence and strength of anomalous signal in data. It was at first surprising to see that the low-resolution limit, $d_{\text{max}}$, played such a significant role in the success of experimental phasing, but upon reflection it is well known that in solvent flattening methods the presence of low-resolution data is important for the definition of the protein envelope and solvent region [123,124]. Other metrics often used by crystallographers to judge the quality of data were discarded through this analysis as they were not deemed useful in answering the specific question asked. However, for some metrics, in particular those describing the precision of unmerged ($R_{\text{merge}}, R_{\text{meas}}$) and merged data ($I/\sigma, CC_{1/2}, R_{p.i.m.}$), the authors were able to confirm long-held views [16,125] regarding which of these best describe the overall data quality of a dataset. In particular, it was found that, especially for merged data, $CC_{1/2}$ was an important feature to describe the data and supports its use as an overall data quality metric. The features found were used to predict whether a dataset can be used to calculate experimental phases or not, i.e. a binary classification problem. Structures for which phases were originally determined using molecular replacement and where experimental setup meant no anomalous signal was produced were assigned to the negative class, ‘0’. If experimental phasing was used when determining the published structure, then the data were labelled to be of the positive class, ‘1’. Although more positive samples were present in the data, the imbalance did not require any additional adjustment and applying stratification when splitting data into training and testing set as well as using appropriate weights during training was sufficient to ensure the predictor was not biased towards the dominant class. In a second round of training only using the six most important features identified, a decision tree with Ada Boost, for general introduction see above, was found as the most stable and best-performing predictor as judged by commonly used assessment metrics. A probability threshold of 80% for class ‘1’ was applied to ensure that predictions for this class had to have high confidence.
To be able to use the predicted probabilities, the predictor was calibrated with some unseen data. Application on some entirely different user data, 24 samples in total, which was not used for training and testing, still produced good results but overall, significantly lower. The accuracy reached in the challenge was 79% with a sensitivity of 64% (finding seven out of 11 positive samples) and a specificity of 92% (finding 12 out of 13 negative samples). This meant that the predictor performed better in correctly identifying samples where experimental phasing as either SAD or MAD was likely to fail. The authors suggested that the most likely reason for the lower performance was a technology gap as the public data used for training the predictor was often acquired on a hardware setup that was the state-of-the-art 6, 12 or more months in the past due to the time needed to solve and publish a structure. The performance during real-life user operation is currently being assessed as the predictor has been integrated into the facility’s data analysis pipelines and will be published in due course.

**Protein crystallizability**

There have been several attempts over the years to create predictive tools for the crystallizability of proteins. One of the earlier implementations, the prediction server **SECRET**, was by Smialowski et al. [55] using a combination of support vector machines and a naïve Bayes classifier. The system was developed using structures determined by either X-ray crystallography or nuclear magnetic resonance (NMR) experiments. The structures were taken from the PDB and additional information from structural genomics work found in the TargetDB ([126]; http://targetdb.pdb.org/) was added after adjustment for the different protein size ranges, which are typically revealed by the two techniques. Physico-chemical properties of a protein that can be derived from its primary sequence were used to create the features for training. As the aim was to predict crystallizability, the negative samples were represented by NMR structures for which no equivalent crystal structures were available and the positive samples were structures solved by X-ray crystallography. More recently, due to high-throughput work in structural genomics initiatives, where data was published in publicly available databases and data mining, the creation of more sophisticated predictive tools has been possible. Here, we look at some more recent developments in the field using support vector machines, random forest and convolutional neural network applications.

The **XtalPred** server developed by Slabinski et al. [110] is one of the most popular and well-known crystallizability prediction servers. It is based on large-scale data analysis of open-source structural genomics results which was used to identify features that can help to distinguish crystallizable from non-crystallizable proteins [127]. Slabinski et al. [110] used a logarithmic opinion pool [128] to combine probabilities for different features into a single score. It used physico-chemical features of proteins to place a sequence into one of five classes of chances of crystallizability: optimal, suboptimal, average, difficult, very difficult. The next generation of **XtalPred** server, now **XtalPred-RF**, involved a substantial change to the underlying algorithm [74]. Besides including new features and larger training and testing sets compared to the first published version, the authors also made use of recent developments of advanced machine learning and AI techniques. The authors explored support vector machines (SVM), artificial neural networks (ANN) and random forest methods. For each system tried, commonly used quality metrics were reported.
SVMs were used in binary classification whereas for ANNs depth, complexity and internally applied algorithms of the network were explored. For the random forest, which was ultimately selected, the maximum number of trees was limited. In all cases, almost all the parameters were kept as defaults. Naturally, the type of data available meant that they were highly imbalanced towards the failure cases, proteins would not crystallize, and undersampling [129] was applied to reduce the number of failure cases. The authors created multiple random forests, each being trained on one of the classes (and hence differently balanced training sets) defined in the earlier implementation as multi-class classification was not directly accessible. To address overfitting, the authors expanded from the data used to develop the first version of XtalPred-RF and looked at feature importance and removed those that were irrelevant for the decision making. Also, sequence similarity was considered, and steps put in place to reduce redundancy. All machine learning methods tried here performed slightly better than the original XtalPred-RF if trained and tested on the data to develop the first version of the predictor, with the random forest coming top. Several of the newly introduced features boosted the predictive performance even though some of them are correlated. The random forest in particular can work with these correlations. Compared to the first version of XtalPred the new RF version setup with additional features improves accuracy from 68% to 74%, specificity from 72% to 78% and sensitivity from 66% to 69%. The higher value achieved for the specificity indicates that the predictor performs better in identifying proteins that are unlikely to crystallize.

Mizianty and Kurgan [61] explored support vector machines in a system named PPCPred to predict the crystallizability for a given protein sequence. They used PepcDB, an extension of TargetDB [126], as data source, which is a further extension to the data used for the development of XtalPred. Rather than predicting an overall likelihood of crystallizability, they give chances of success at different stages along the path from construct to crystal. After careful filtering and data assessment, three datasets of non-crystallizable proteins were created with each set representing a different step in the crystallization process where failure occurred, and for each set a separate predictor was trained. In total, four class labels were annotated, three for the different stages of crystallization failure and one for success. At each stage, a binary classification was carried out. As with the new XtalPred version, the training data had more samples of the failure than the success cases. For the training, the authors had selected similar features as Jahandideh et al. [74] and combined them into a single, numerical feature vector. For each of the possible crystallization outcomes an SVM was trained with fivefold cross-validation and the predictions from all four models were aggregated into a final result. Three different kernels were explored: radial basis function, polynomial (which also covers the linear kernel) and sigmoid. In total, 12 models were computed and assessed; the 4 possible outcomes multiplied with 3 different kernels selected above. Several quality metrics are used to assess performance depending on what is being predicted: Matthews correlation coefficient (MCC; [130]), accuracy, sensitivity, specificity and receiver operator characteristics for the area under the curve. A final overall accuracy and mean MCC was then calculated after aggregating the results for the four classes. Bootstrapping and Student’s t-test were used to measure statistical significance. Pearson correlation coefficient and the cross-validation method were combined to eliminate large numbers of correlated features so that for each of the final predictors, approximately the same number of elements were in the feature vector. Of the 12 models available, the best performing for a particular class outcome had their results combined,
whereby at each step a probability threshold was applied. The overall performance achieved was an improvement to existing methods but accuracy remained similar. For the predictions to be able to crystallize diffraction quality crystals, the achieved accuracy, specificity and sensitivity were 77%, 85% and 61%, respectively. As with XtalPred, this means that the predictor performs better in identifying proteins that will fail to produce diffraction quality crystals. Considering that data was not assessed and balanced accounting for the different classes, reconsidering the class distribution may be a first step to further improve performance. This method is the closest in terms of performance to the above-mentioned XtalPred server and compared to the latter’s original implementation, Mizianty and KurGAN [62] showed improved prediction success. The newer version of XtalPred-RF from 2014 has not been assessed.

Wang et al. [63] used support vector machines in a stacked manner where the output of one SVM is the input to another. More importantly, they produced a well-curated dataset from structural genomics data, which has not only found use in benchmarking but enabled other groups to develop new applications to predict the chance of protein crystallizability. This was of major importance, as, over time, with new developments in laboratory equipment and improvements in experimental techniques previously annotated failure cases (proteins that were thought not to be crystallizable) could now be purified and crystallized. As a result, previously developed prediction tools became unstable and produced poor results when presented with new samples. Looking at the implementations by other groups in the past, Wang et al. [63] created PredPPCrys (Prediction of Procedure Propensity for protein Crystallization) to not just predict the crystallizability in a single class, but rather give chances of success for the individual steps on the way to a crystal structure. Their algorithm made use of a combination of features identified in previous work, i.e. amino acid indices, types, compositions, physico-chemical properties, predicted structural features generated from the sequence, the PROFEAT server [131] and other bioinformatics tools. As with most machine learning projects, a significant amount of time and resources were used by the authors to curate the data in terms of sample completeness, ensuring non-redundancy of the samples, defining the classes to be predicted and determining the most important features for each of the different classes. Using common quality metrics as have been introduced above, the predictor created by Wang et al. [63] outperformed many of the past implementations on various levels. With a specificity of 76% and a sensitivity of 75% the predictor performs similarly well on both cases when a protein may crystallize or not. The accuracy achieved is 76% which is similar to what was found for XtalPred-RF and PPCPred.

The most recent approach to predict protein crystallizability is by Alavi and Ascher [75], DHS-Crystallize. Like the previous two methods, the focus here was to analyse a protein sequence regarding physico-chemical, sequence-based and functional features to estimate how likely this will result in a protein crystal. The authors used a deep convolutional neural network (CNN) to extract features from the sequence which were then combined with structural and physico-chemical features. The higher performance achieved in this system was attributed to the automatically identified features in the sequence which allowed for a better description of the problem than traditional, hand-crafted features. The system explored here was based on some previous work that solely relied on a deep CNN and the raw protein sequence [84]. The predictions were made for a binary classification problem, ‘does crystallize’ or ‘does not’. Training was done with publicly, curated data [63] and
two independent blind test sets were used to assess performance. As with the other two applications above, filtering was carried out regarding sequence redundancy in the different datasets. The authors used the same network architecture as published by Elbasir et al. [84] and extracted the features from the output of the fully connected layer. The final connected layer produced 250 features and the results for 10 repeats were concatenated. 21 physico-chemical features were derived from the protein sequence using the Protlearn software [132]. The actual binary classification was then conducted using the XGBoost algorithm [133], an ensemble decision tree method. To improve the predictors performance, dimensionality reduction was carried out using PCA to reduce the feature vector to a quarter of its initial size when coming from feature extraction. The data used in training were balanced by applying appropriate weights. The XGBoost based predictor achieved better performance on the test set as well as the two blind sets compared to all the other methods included in this study as judged by commonly used quality metrics. The results presented by Alavi and Ascher [75] give an initial idea of the design of a future system. There were several elements in connection with the dimensionality reduction step and the XGBoost training where more thorough exploration of parameters may result in even better performance. This has already been mentioned by Alavi and Ascher [75]. Depending on the test set this CNN-based predictor reaches accuracies between 77% and 89% with precision between 81% and 96% and recall between 67% and 92%. No information is given about performance on negative class samples in form of specificity. This is a significant improvement in accuracy by ∼10% compared to previous methods.

**Crystallization outcome classification**

Large screening campaigns to identify and optimize conditions for novel structures as well as producing large numbers of crystals of consistent quality for small molecule soaking experiments are routinely done in structural genomics and pharmaceutical settings. Automated crystallization platforms are often used to facilitate such high throughput experiments by ensuring safe and managed storage with automated imaging of the crystallization trials at set time points. If the crystallization conditions for a target are well established, then results can usually be expected within a few days and crystals can be identified easily due to their size. However, when screening a large number of conditions to crystallize a novel protein, it can be challenging to manually assess the large number of images produced. To support crystallographers in this assessment and to make most out of automated crystallization platforms, image recognition techniques have been developed as, for example, by the MArchine Recognition of Crystallization Outcomes initiative (MARCO; [20]).

For MARCO the training images were collected on different imaging systems at a hand full of high-throughput labs. Each system had a specific setup defined by the group so that the total number of images was reasonably diverse. The images were labelled by using existing scoring systems and collapsing them to a common set of four labels. By using the existing scores, a mix of people were involved in the labelling, reducing the likelihood of bias towards one person’s scoring strategy. After adjusting some of the labels from initial trials and cleaning the data, a deep convolutional neural network was created based on the work of LeCun et al. [134,135] and Rawat and Wang [136]. Some parts of the architecture...
were directly taken from Szegedy et al. [137]. After adjusting some parts of the network architecture to suit the problem and training with the large and diverse image set, the classifiers performance surpassed that of already available academic or commercial systems when tested by the same groups that contributed images. The trained model is open-source and can be run locally by any other laboratory. Performance for these new users with a new, specific local setup may however be lower compared to that reported, presumably due to the presence of specific characteristics of the local setup. Such a difference has been observed while the system has been in use at the crystallization facility at Diamond Light Source and a more personalized system has been developed to be implemented at beamline VMX-i to address deficiencies (Olly King, personal communication).

Before the success of convolutional neural networks and the striking developments in computer vision, earlier algorithms explored other areas of machine learning. Cumbaa et al. [48] and Cumbaa and Jurasica [49] as well as Saitoh et al. [50] used linear discriminant analysis to identify which images of drops in crystallization trials contained actual protein crystals. In these three approaches, the manually labelled images provided the ground truth and the trained algorithms were usually able to correctly identify around 85% of these images. Decision trees were explored by Bern et al. [72] and Liu et al. [73]. The latter adds a boosting algorithm to the basic decision tree which allows the usage of more and often marginal features in the decision process. Buchala and Wilson [51] present ALICE (Analysis of Images from Crystallization Experiments) which is based on previous work and combines an object-based method [138] with wavelet transforms [139] with the final classification being carried out by an SVM. Pan et al. [64] used an SVM-based system to classify images of crystallization trials. Although taking some of the workload of a crystallographer, their system was plagued by a high false-positive rate of nearly 40%. A self-organising net was proposed by Spraggon et al. [85] and implemented in their package CEEP (Crystal Experiment Evaluation Program). The algorithms presented here substantially reduce a crystallographer’s time spent examining and scoring crystallization trials. The main problem for these techniques remains that they are very specific to the local experimental setup that was used to produce the crystallization trials and acquire the drop images. Even with a state-of-the-art algorithm used in MARCO [20] this problem persists.

**Protein modelling**

A very detailed review of the state-of-the-art developments and some historic lead-up regarding protein modelling is given by Gao et al. [140]. They focus on deep learning applications that have begun to replace the more traditional statistics and machine learning methods, covering structure prediction, protein folding, protein design and molecular dynamics simulation.

Until a few years ago, most structure prediction implementations would focus on structure optimization methods combined with some information from evolutionary coupling analysis (ECD [141]). The assumption here was that two amino acid residues that are in close contact with each other in a three-dimensional protein structure will co-evolve so as not to disrupt the integrity of the protein structure. RaptorX [88] and its implementation of deep learning to predict the distances between pairs of residues was a major breakthrough, outperformed ECD and paved the way for more sophisticated applications as will be discussed below.
In [140] the authors discuss different deep learning and neural network types that have proven successful in scientific modelling competitions. They give details about the inner structures of these deep learning algorithms and explain what procedures and algorithms have been used to turn a protein's amino acid sequence and additional protein descriptors into a 3D structure. A learnable feature vector is created, often using algorithms from natural language processing such as Word2Vec or Doc2Vec as was first proposed by Asgari and Mofrad [142]. The summary provided by Gao et al. [140] gives a good foundation to understand why, when combining all the advances and developments the algorithm, *AlphaFold* [89,90] designed by Google's DeepMind, was so successful.

As the result of the unprecedented advances in the field of protein structure prediction thanks to the work of the developers of *RaptorX* [88] and *AlphaFold* ([89,90]; see below) another implementation was developed. Yang et al. [91] created a new method, transform-restrained Rosetta (*trRosetta*), that incorporated the ideas of [88] and [89,90] and generally deep residual convolutional neural networks ideas as were used by all the top-scoring groups during the 13th Critical Assessment of Protein Structure Prediction (CASP13; [143]) competition in 2018. These deep learning algorithms were combined with coevolutionary coupling features determined from multiple sequence alignments (MSAs) with the output being distances that yield structures through direct optimization. Yang et al. [91] extended the CASP13 ideas by choosing a Rosetta-based optimization that uses a Rosetta energy function, which allows for direct model building. A total of six geometric descriptors were predicted to describe the relative orientation of two residues to each other. Similar to [89,90] the predicted distances and orientations are used in a minimization function and their probability distributions are converted into inter residue interaction potentials. Those potentials were then combined with a Rosetta centroid level (coarse-grain) energy function [144] to create folded structures. The model with the lowest energy and the most built atoms was selected as final model.

*AlphaFold* developed by Google’s DeepMind [89, 90] is a structure prediction tool that uses deep learning and was one of the contestants during CASP13 competition. They trained a complex 2D dilated convolutional residual network (whose basic design is common in 2D image recognition) to predict distances between pairs of amino acids based on a protein’s sequence. Backbone torsion angles and relative accessible surface for each residue were also predicted. Furthermore, based on the complexity of the network the secondary structure was predicted based on DSSP [145] class labels. The data used to predict the pairwise distances were extracted from PDB structures after creating MSAs which produced several hundred features. The structures were curated to ensure non-redundancy and during train-test splitting homologous structures were kept together in the same set. So rather than using fragments of homologous structures for training a predictor, which is usually done and often combined with contact predictions, they used the smaller-sized distance pairs. To avoid overfitting, MSAs were subjected to data augmentation by subsampling the alignments and the PDB coordinates had noise added, resulting in variation in the target distances, which, combined, increased the number of training samples. In a second step, the derived covariance information was used in a gradient descent procedure, using L-BFGS [146] as minimizer, to create a potential of mean force which was then used to describe the shape of a protein. At the time of the CASP13 competition DeepMind’s *AlphaFold* considerably advanced the entire structure prediction field for certain challenges, the free modelling class in particular. In the free modelling class, the provided
targets were novel structures for which no known homologous structures were available. *AlphaFold* achieved the highest scores for correctly modelling the backbone trace and hence overall fold of given target proteins. Side chains were a different matter. Using the same algorithm, which did not include structural information from homologues for training, the group still achieved high scores in the template-based modelling category. The entire system relied, however, on the accuracy of the distance predictions.

In 2020 the next iteration, CASP14 ([https://predictioncenter.org/casp14/index.cgi](https://predictioncenter.org/casp14/index.cgi)), saw DeepMind entering again, this time with an entirely new system, *AlphaFold2* [111]. Looking at the few details provided in the competition’s book of abstracts and including some general knowledge and ideas in the field of AI we try to identify some components of this new system. The input data for *AlphaFold2* were MSAs and template hits from homologues, and the output is a fully folded protein structure as defined by distances, torsion angles and atomic coordinates. The predictor itself was based upon convolutional neural networks (CNN) which capture local details either between pixels in 2D or voxels in 3D. CNNs, however, lack more global information, i.e. long-distance interactions in a 3D structure, which need to be found from a 1D sequence. This challenge is close to natural language processing, where a system needs to be able to keep track of what happened at the beginning and middle of a sentence or paragraph (or a particular amino acid in a sequence) and how this affects the end or conclusion (or the position and interaction in a 3D structure). A likely choice of CNN here would be an attention-based network [147] that aims to find the most important connections between pairs of residues and between a residue and the corresponding one in other sequences in the MSA. At the same time, the predictor ignores irrelevant sequences in the alignment and identifies long-range interactions between amino acid pairs in a more holistic way than through co-evolution between pairs of amino acids in a structure. A transformer, probably similar to generative pre-trained transformer-3 (GPT-3; [148]) or deep bidirectional encoder transformer (BERT; [149]), could be used to update the protein backbone and build side chains. The entire process of looking at an MSA, working out connections and building a model was iterative and repeated a few times over to reach convergence. Finally, the structure was then ‘settled’ using coordinate-restrained gradient descent with an Amber force field [150]. The predictor did extremely well in the competition in all categories, coming top in the free-modelling competition and achieving high scores in the template-based challenges. Out of a total 43 targets, *AlphaFold2* achieved the highest score for 37 of these. For five samples the results were comparable to the other groups and for one the submitted structure was worse. It should be noted however, that the system was very good in predicting a protein crystal structure given a MSA, considering that the training data and features stem from the PDB. The PDB is largely dominated by protein structures solved from protein crystals using X-ray diffraction and is biased towards proteins that are crystallizable, i.e. there is a sparse representation of structures being solved by nuclear magnetic resonance or cryo-electron microscopy experiments, membrane proteins, intrinsically disordered proteins and large multi-protein complexes. This may explain why it is challenging, even for *AlphaFold2*, to correctly predict solvent-exposed protein surfaces and loop regions. Such areas are known to be highly flexible which in turn hampers them from forming close and stabilizing contacts to create an ordered crystal lattice. A high level of order in a protein crystal is necessary to get sufficient resolution to build a correct 3D structure. Unfortunately, highly flexible areas are often the most interesting ones from a biological
function perspective and since structure and function are inextricably linked, the ability to correctly predict these more disordered structural regions, or indeed the nature of the disorder, will be critical to furthering our understanding of biological function. The same applies to protein-protein interactions, which again can involve highly flexible/mobile surface features of a protein (or to continue with natural language processing to keep track of different characters in a story and how they affect each other over several chapters or even books in a series). Another challenge is the placement of ligand molecules in biologically and chemically relevant interactions to boost drug design or to predict function and mechanism and identify the active site of novel folds for which any type of experimental data is sparse. So, in all, AlphaFold2 is remarkably good at predicting protein structures with a strong bias towards well-ordered structures like those largely represented in the PDB.

Based on the published details for AlphaFold and AlphaFold2 and general domain-specific knowledge, the Baker lab has recently published a new structure prediction algorithm, RoseTTAFold [112]. Baek et al. [112] explore a model with three parallel tracks, where 1D information from the amino acid sequence is combined with 2D distance maps and 3D atomic coordinates. The three tracks talk to each other and exchange information to optimize weights and parameters in the system. To assess the performance of the system blindly, the server has been taking part in the CAMEO experiment [151] and so far outperformed all other participants. The algorithm aims to close the gap, seen in CASP14, that exists between academic groups and DeepMind and make the method available to the scientific community. Unfortunately, DeepMind is a private company with proprietary interests and the AlphaFold/2 developers may not be able to fully disclose all details.

Concluding remarks

In this review, we have provided a brief history and given a targeted introduction to some of the key concepts, ideas and algorithms used in machine learning and artificial intelligence and reviewed some of their applications in the field of macromolecular crystallography. Machine learning and artificial intelligence are fast-moving areas of research greatly accelerated by new developments in computing technology and the availability of open-source software: keeping pace with new and emerging methods is in itself a challenge. This review is intended to deliver some starting points for anyone in the structural biology field who might be curious about applying machine learning to their problem. As many research institutions support open data, there are numerous publicly accessible scientific databases and experimental data are produced at an ever-increasing rate. As such, data science, machine learning and artificial intelligence will be new tools that can be employed to explore interesting questions from different points of view. The various sources of information can be combined into highly complex sets of data that do not necessarily serve a single purpose, for example, to solely solve a structure, but more broadly describe a biological problem. For example, information from a multi-sequence alignment can be combined with contact predictions, secondary structure predictions, protein folding, functional annotations based on phylogeny and atomic coordinates with their positional deviations from an ensemble of homologues. A combination of such data may be used to deduct a protein’s structure and function if crystallization has not succeeded,
and biochemical assays prove difficult because the protein is unstable. As with ensemble predictors, where weak predictions are combined to give a strong answer, if any single piece of information is of low quality or shows a weak signal combining them may allow conclusions with more certainty to be drawn. Additionally, if some small molecules can be docked and their geometry can be provided then an AI system may be able to identify a reaction mechanism in an active site and whether a potential drug may have any effect. Molecular dynamics simulation may then be used to assess how trustworthy such a prediction is. Ideas for such systems are already being trialled [152,153]. Machine learning and AI offer a way to explore not just protein structures but a much larger space of data that may never be realistically accessible through experiments. This will allow for a good guess as to what the ground truth may be and, with experience and time, will become ever closer to the perfect solution. Finally, by using data-driven approaches there is a real opportunity to challenge and expand established crystallographic analysis methods. By creating expert systems that encapsulate decades of experience in the assessment of the quality and usefulness of diffraction data from the many metrics available during analysis, X-ray beamlines will themselves be able to make decisions on the best steps to take towards successful crystallographic structure determination.

**Acknowledgements**

We thank Sir Dave Stuart, FRS, and David Hall for fruitful discussions and comments when writing this review.

**Disclosure statement**

No potential conflict of interest was reported by the author(s).

**Funding**

This work was supported by BBSRC: [Grant Number BB/S006699/1].

**Notes on contributors**

*Gwyndaf Evans* was awarded his PhD in Physics from the University of Warwick in 1994 for his work performed at the European Molecular Biology Laboratory in Hamburg on the application of anomalous scattering in macromolecular crystallography using synchrotron radiation. After Post-Doctoral positions at the Advance Photon Source in Chicago, the MRC Laboratory of Molecular Biology and Global Phasing Ltd (Cambridge, UK), he joined Diamond Light Source on the Harwell Campus in 2004. In 2020, he took up a joint position with the Rosalind Franklin Institute (Oxfordshire, UK) as Head of Technology and became Deputy Director of Life Sciences at Diamond Light Source. Dr Evans is a leading expert in the use of synchrotron radiation for structural biology and has research groups developing instrumentation, software and methods in X-ray and electron diffraction and imaging at the interface of measurement and analysis.
Melanie Vollmar did her doctoral studies in structural biology at the University of Düsseldorf, Germany, and was awarded a Dr. rer. nat. in 2009. After a Post-Doctoral position at the Structural Genomics Consortium at the University of Oxford, UK, followed by another at the University of Manchester, UK, Dr Vollmar joined Diamond Light Source in 2014. Her work at Diamond combines her strong expertise in protein crystallography and structural biology with data science, machine learning and statistics.

ORCID

Melanie Vollmar http://orcid.org/0000-0002-9162-9159
Gwyndaf Evans http://orcid.org/0000-0002-6079-2201

References

[1] Yokoyama S, Matsuo Y, Hirota H, et al. Structural genomics projects in Japan. Prog Biophys Mol Biol. 2000;73:363–376.
[2] Cassetta A, Deacon AM, Ealick SE, et al. Development of instrumentation and methods for MAD and structural genomics at the SRS, ESRF, CHESS and Elettra facilities. J Synchrotron Radiat. 1999;6:822–833.
[3] Weigelt J. Structural genomics – impact on biomedicine and drug discovery. Exp Cell Res. 2010;316:1332–1338.
[4] Joachimiak A. High-throughput crystallography for structural genomics. Curr Opin Struc Biol. 2009;19:573–584.
[5] Delagérière S, Brenchereau P, Launer L, et al. ISPyb: an information management system for synchrotron macromolecular crystallography. Bioinformatics. 2011;27:3186–3192.
[6] Berman HM, Westbrook J, Feng Z, et al. The protein data bank. Nucleic Acids Res. 2000;28:235–242.
[7] Gelski M. Automation and remote synchrotron data collection. Acta Phys Pol A. 2008;114:331–338.
[8] Sanchez-Weatherby J, Sandy J, Mikolajek H, et al. VMXi: a fully automated, fully remote, high-flux in situ macromolecular crystallography beamline. J Synchrotron Radiat. 2019;26:291–301.
[9] Bowler MW, Nurizzo D, Barrett R, et al. MASSIF-1: a beamline dedicated to the fully automatic characterization and data collection from crystals of biological macromolecules. J Synchrotron Radiat. 2015;22:1540–1547.
[10] Winter G. Xia2: an expert system for macromolecular crystallographic data reduction. J Appl Crystallogr. 2010;43:186–190.
[11] Winter G, McAuley KE. Automated data collection for macromolecular crystallography. Methods. 2011;55:81–93.
[12] Rao R. Phys Today. 2020. Available from:https://physicstoday.scitation.org/do/10.1063/PT.6.2.20200925a/full/.
[13] Keegan RM, Winn MD. MrBUMP: an automated pipeline for molecular replacement. Acta Crystallogr D. 2008;D64:119–124.
[14] Vonrhein C, Blanc E, Roversi P, et al. Automated structure solution with autoSHARP. Methods Mol Biol. 2007;364:215–230.
[15] Diederichs K, Karplus PA. Improved R-factors for diffraction data analysis in macromolecular crystallography. Nat Struct Biol. 1997;4:269–275.
[16] Weiss MS, Hilgenfeld R. On the use of the merging R factor as a quality indicator for X-ray data. J Appl Crystallogr. 1997;30:203–205.
[17] Weiss M. Global indicators of X-ray data quality. J Appl Crystallogr. 2001;34:130–135.
[18] Evans P. Scaling and assessment of data quality. Acta Crystallogr D. 2006;D62:72–82.
[19] Yang J, Anishchenko I, Park H, et al. Improved protein structure prediction using predicted interresidue orientations. Proc Natl Acad Sci USA. 2020;117:1496–1503.

[20] Bruno AE, Charbonneau P, Newman J, et al. Classification of crystallization outcomes using deep convolutional neural networks. PLoS ONE. 2018;13:e0198883.

[21] Kryshtafovych A, Monastyrskyy B, Fidelis K. CASP11 statistics and prediction center evaluation system. Proteins. 2016;84:15–19.

[22] Kohavi R, Provost F. Glossary of terms. Mach Learn. 1998;30:271–274.

[23] Bishop CM. Pattern recognition and machine learning. New York: Springer; 2006.

[24] McCulloch S, Pitts W. A logical calculus of the ideas immanent in nervous activity. B Math Biophys. 1943;5:115–133.

[25] Hebb DO. The organization of behaviour. A neurophysiological theory. New York: John Wiley & Sons, Inc.; 1949.

[26] Hubel DH, Wiesel TN. Receptive fields of single neurons in the cat’s striate cortex. J Physiol. 1959;148:574–591.

[27] Hubel DH, Wiesel TN. Receptive fields, binocular interaction and functional architecture in the cat’s visual cortex. J Physiol. 1962;160:106–154.

[28] Turing A. Computing machinery and intelligence. Mind. 1950;LIX:433–460.

[29] Minsky M. A neural-Analogue calculator based upon a probability model of reinforcement. Cambridge (Massachusetts): Harvard University Psychological Laboratories; 1952.

[30] McCarthy J, Minsky ML, Rochester N, et al. A proposal for the Dartmouth summer research project on artificial intelligence. AI Mag. 1955;27:12–14.

[31] Samuel AL. Some studies in machine learning using the game of checkers. IBM J Res Dev. 1959;3:210–229.

[32] Samuel AL. Some studies in machine learning using the game of checkers. II - recent progress. IBM J Res Dev. 1967;11:601–617.

[33] Rosenblatt F. The Perceptron: A Perceiving and Recognizing Automaton. Report: 85–60. -1, Cornell Aeronautical Laboratory, Buffalo, New York, 1957.

[34] Cover T, Hart P. Nearest neighbor pattern classification. IEEE T Inform Theory. 1967;13:21–27.

[35] Fukushima K. Neural network model for a mechanism of pattern recognition unaffected by shift in position - neocognitron. Trans IECE. 1979;J62-A:658–665.

[36] Fukushima K. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. Biol Cybern. 1980;36:193–202.

[37] Linnaimaa S. The representation of the cumulative rounding error of an algorithm as a Taylor expansion of the local rounding errors [Master’s Thesis (in Finnish). ”, University of Helsinki, 1970.

[38] Schapire RE. The strength of weak learnability. Mach Learn. 1990;5:197–227.

[39] Hochreiter S, Schmidhuber J. Long short-term memory. Neural Comput. 1997;9:1735–1780.

[40] Pearson KLIII. On lines and planes of closest fit to systems of points in space. Philos Mag. 1901;2:559–572.

[41] Wold S, Sjöström M, Eriksson L. PLS-regression: a basic tool of chemometrics. Chemometr Intell Lab. 2001;58:109–130.

[42] Kruskal JB. Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. Psychometrika. 1964;29:1–27.

[43] Borg I, Groenen P. Modern Multidimensional scaling: theory and applications. New York: Springer; 2005.

[44] Fisher RA. The use of multiple measurements in taxonomic problems. Ann Eugen. 1936;7:466–475.

[45] McLachlan GJ. Discriminant analysis and statistical pattern recognition. In: Wiley series in probability and statistics. A Wiley interscience publication. New York: John Wiley & Sons, Inc.; 2004.

[46] Pereira J, Lamzin VS. A distance geometry-based description and validation of protein main-chain conformation. IUCrJ. 2017;4:657–670.
[47] Hattne J, Lamzin VS. Pattern recognition-based detection of planar objects in 3D electron density maps. Acta Crystallogr D. 2008;D64:834–842.
[48] Cumbaa CA, Lauricella A, Fehrman N, et al. Automatic classification of sub-microlitre protein-crystallization trials in 1536-well plates. Acta Crystallogr D. 2003;D59:1619–1627.
[49] Cumbaa C, Jurisica I. Automatic classification and pattern discovery in high-throughput protein crystallization trials. J Struct Funct Genomics. 2005;6:195–202.
[50] Saitoh K, Kawabata K, Asama H, et al. Evaluation of protein crystallization states based on texture information derived from greyscale images. Acta Crystallogr D. 2005;D61:873–880.
[51] Buchala S, Wilson JC. Improved classification of crystallization images using data fusion and multiple classifiers. Acta Crystallogr D. 2008;D64:823–833.
[52] Manning CD, Raghavan P, Schütze H. Introduction to information retrieval. New York: Cambridge University Press; 2008.
[53] Pearl J. Bayesian Networks: A Model of Self-Activated Memory for Evidential Reasoning. UCLA Technical Report CSD-850017. Proceedings of the 7th Conference of the Cognitive Science Society, University of California, Irvine, CA. 1985:329–334.
[54] Bayes Rev T. An essay towards solving a problem in the doctrine of chances. Philos T R Soc Lond. 1763;53:370–418. Reprinted in Biometrika, 1958;45:293-315.
[55] Santosa F, Symes WW. Linear inversion of band-limited reflection seismograms. SIAM J Sci Stat Comput. 1986;7:1307–1330.
[56] Tibshirani R. Regression shrinkage and selection via the lasso. J R Stat Soc. 1996;58:267–288.
[57] Altman NS. An introduction to kernel and nearest-neighbor nonparametric regression. Am Stat. 1992;46:175–185.
[58] Fix E, Hodges JL. Discriminatory analysis. Nonparametric discrimination: Consistency properties. Report Number 4, Project Number 21-49-004. USAF School of Aviation Medicine, Randolph Field, Texas, 1951.
[59] Cortes C V. VN. support-vector networks. Mach Learn. 1995;20:273–797.
[60] Mizianty MJ, Kurgan L. Sequence-based prediction of protein crystallization, purification and production propensity. Bioinformatics. 2011;27:i24–i33.
[61] Wang H, Wang M, Tan H, et al. PredPPCrys: accurate prediction of sequence cloning, protein production, purification and crystallization propensity from protein sequences using multi-step heterogeneous feature fusion and selection. PloS ONE. 2014;9:e105902.
[62] Pan S, Shavit G, Penas-Centeno M, et al. Automated classification of protein crystallization images using support vector machines with scale-invariant texture and gabor features. Acta Crystallogr D. 2006;D62:271–279.
[63] Chojnowski G, Pereira J, Lamzin VS. Sequence assignment for low-resolution modelling of protein crystal structures. Acta Crystallogr D. 2019;D75:753–763.
[64] Breiman L, Friedman JH, Olshen RA, et al. Classification and regression trees. Wadsworth & Brooks/Cole, Advanced Books & Software Monterey, 1984.
[65] Ho TK. Random decision forests). ICDAR ’95 Proceedings of the Third International Conference on Document Analysis and Recognition; 1995:1:278.
[66] Breiman L. Bagging predictors. Mach Learn. 1996;24:123–140.
[67] Friedman JH. Greedy function approximation: A Gradient Boosting machine. Ann Stat. 2001;29:1189–1232.
[68] Mason L, Baxter J, Bartlett L, et al. Boosting Algorithms as Gradient Descent.). NIPS’99: Proceedings of the 12th International Conference on Neural Information Processing Systems; 1999:512–518.
[69] Geurts P, Ernst D, Wehenkel L. Extremely randomized trees. Mach Learn. 2006;63:3–42.
[70] Bern M, Goldberg D, Stevens RC, et al. Automatic classification of protein crystallization images using a curve-tracking algorithm. J Appl Cryst. 2004;37:279–287.
[73] Liu R, Freund Y, Spraggon G. Image-based crystal detection: a machine learning approach. Acta Crystallogr D. 2008;D64:1187–1195.

[74] Jahandideh S, Jaroszewski L, Godzika A. Improving the chances of successful protein structure determination with a random forest classifier. Acta Crystallogr D. 2014;D70:627–635.

[75] Alavi A, Ascher DB. DHS-Crystallize: Deep-Hybrid-Sequence based method for predicting protein Crystallization. Posted: [cited 2020 Nov 13] Available from:https://www.biorxiv.org/content/10.1101/2020.11.13.381301v1.

[76] Vollmar M, Parkhurst JM, Jaques D, et al. The predictive power of data-processing statistics. IUCrJ. 2020;7:342–354.

[77] Bishop CM. Neural networks for pattern recognition. New York: Oxford University Press, Inc; 1995.

[78] Rumelhart DE, Hinton GE, Williams RJ. Learning representations by back-propagation errors. Nature. 1986;323:533–536.

[79] Hopfield JJ. Neural networks and physical systems with emergent collective computational abilities. Proc Natl Acad Sci USA. 1982;79:2554–2558.

[80] Hinton GE, Salakhutdinov R. A better way to pretrain deep Boltzmann machines). NIPS’12: Proceedings of the 25th International Conference on Neural Information Processing Systems; 2012.:2447–2455.

[81] Hinton GE, Osindero S, Teh Y-W. A fast learning algorithm for deep believe nets. Neural Comput. 2006;18:1527–1554.

[82] Watkins CJCH. Phd thesis [Learning from Delayed Rewards, King’s College], Cambridge, UK, 1989.

[83] Bond PS, Wilson KS, Cowtan KD. Predicting protein model correctness in Coot using machine learning. Acta Crystallogr D. 2020;D76:713–723.

[84] Elbasir A, Moovarkumudalvan B, Kunji K, et al. Deepcrystal: a deep learning framework for sequence-based protein crystallization prediction. Bioinformatics. 2019;35:2216–2225.

[85] Spraggon G, Lesley SA, Kreusch A, et al. Computational analysis of crystallization trials. Acta Crystallogr D. 2002;D58:1915–1923.

[86] Souza A, Oliveira LB, Hollatz S, et al. DeepFreak: Learning Crystallography Diffraction Patterns with Automated Machine Learning. Posted: April 26, 2019. Available from:https://arxiv.org/pdf/1904.11834.pdf.

[87] Ke T-W, Brewster AS, Yu SX, et al. A convolutional neural network-based screening tool for X-ray serial crystallography. J Synchrotron Radiat. 2018;25:655–670.

[88] Wang S, Sun S, Li Z, et al. Accurate De novo prediction of protein contact Map by ultra-deep learning model. PLoS Comput Biol. 2017;13:e1005324.

[89] Senior AW, Evans R, Jumper J, et al. Protein structure prediction using multiple deep neural networks in the 13th Critical Assessment of protein structure prediction (CASP13). Proteins. 2019;87:1141–1148.

[90] Senior AW, Evans R, Jumper J, et al. Improved protein structure prediction using potentials from deep learning. Nature. 2020;577:706–710.

[91] Yang J, Anishchenko I, Park H, et al. Improved protein structure prediction using predicted interresidue orientations. Proc Natl Acad Sci USA. 2020;117:1496–1503.

[92] Redmon J, Divvala S, Girshick R, et al. You Only Look Once: Unified, Real-Time Object Detection. Posted: June 8, 2015. Available from:https://arxiv.org/pdf/1506.02640.pdf.

[93] Scheres SHW. RELION: implementation of a Bayesian approach to cryo-EM structure determination. J Struct Biol. 2012;176:212–218.

[94] Tufféry S. Data Mining and statistics for decision making. Chichester: Wiley; 2011.

[95] Zhang KYJ, Main P. Histogram matching as a new density modification technique for phase refinement and extension of protein molecules. Acta Crystallogr A. 1990;A46:41–46.
[98] Jiang M, He H, Cheng Y, et al. Resolution dependence of an Ab initio phasing method in protein X-ray crystallography. Crystals (Basel). 2018;8:156–173.

[99] de Smith MJ. Statistical analysis handbook. Edinburgh: The Winchelsea Press; 2018.

[100] Demiralp C, Haas PJ, Parthasarathy S, et al. Foresight: rapid data exploration through guideposts. Posted: September 29, 2017. Available from: https://arxiv.org/pdf/1709.10513.pdf.

[101] Guyon I, Elisseeff A. An introduction to variable and feature selection. JMLR. 2003;3:1157–1182.

[102] Dodge S, Karam L. A study and comparison of human and deep learning recognition performance under visual distortion. Posted: [Cited 2017 May 6]. Available from: https://arxiv.org/pdf/1705.02498.pdf.

[103] Hanley JA, McNeil BJ. A method of comparing the areas under receiver operating characteristic curves derived from the same cases. Radiology. 1983;148:839–843.

[104] Gosset WS. The probable error of a mean. Biometrika. 1908;6:1–25.

[105] Efron B. Bootstrap methods: another look at the jackknife. Ann Stat. 1979;7:1–26.

[106] Batista GEAPA, Prati RC, Monard MC. A Study of the behavior of several methods for balancing machine learning training data. ACM Sigkdd Explorations Newsletter. 2004;6:20–29.

[107] Allen DM. The relationship between variable Selection and data augmentation and a method for prediction. Technometrics. 1974;16:125–127.

[108] Stone M. Cross-Validatory choice and assessment of statistical predictions. J R Stat Soc B. 1974;36:111–147.

[109] Claesen M, De Moor B. Hyperparameter search in machine learning. Posted [cited 2015 Feb 7]. https://arxiv.org/pdf/1502.02127.pdf.

[110] Slabinski L, Jaroszewski L, Rychlewski L, et al. XtalPred: a web server for prediction of protein crystallizability. Bioinformatics. 2007;23:3403–3405.

[111] Jumper J, Evans R, Pritzel A, et al. Highly accurate protein structure prediction with AlphaFold. Nature. 2021;596:583–589.

[112] Baek M, DiMaio F, Anishchenko I, et al. Accurate prediction of protein structure and interactions using a 3-track network. Science. 2021;373:871–876.

[113] Krizhevsky A, Sutskever I, Hinton GE. Imagenet classification with deep convolutional neural networks. NIPS; 25, Proceedings of the 12th International Conference on Neural Information Processing Systems; 2012:1097–1105

[114] Maia FRNC. The Coherent X-ray Imaging Data bank. Nat Methods. 2012;9:854–855.

[115] Jarrett K, Kavukcuoglu K, Ranzato MA, et al. What is the best multi-stage architecture for object recognition?). Proceedings of the 2009 IEEE 12th International Conference on Computer Vision; 2009:2146-2153.

[116] Winter G, Waterman DG, Parkhurst JM, et al. DIALS: implementation and evaluation of a new integration package. Acta Crystallogr D. 2018;D74:85–97.

[117] Emsley P, Cowtan K. Coot: model-building tools for molecular graphics. Acta Crystallogr D. 2004;D60:2126–2132.

[118] Emsley P, Lohkamp B, Scott WG, et al. Features and development of coot. Acta Crystallogr D. 2010;D66:486–501.

[119] Cowtan K. The Buccaneer software for automated model building. I. Tracing protein chains. Acta Crystallogr D. 2006;D62:1002–1011.

[120] Cowtan K. Fitting molecular fragments into electron density. Acta Crystallogr D. 2008;D64:83–89.

[121] Lovell SC, Word JM, Richardson JS, et al. The penultimate rotamer library. Proteins. 2000;40:389–408.

[122] Ramachandran GN, Ramakrishnan C, Sasisekharan V. Stereochemistry of polypeptide chain configuration. J Mol Biol. 1963;7:95–99.

[123] Wang BC. Resolution of phase ambiguity in macromolecular crystallography. Methods Enzymol. 1985;115:90–112.

[124] Leslie AGW. A reciprocal-space method for calculating a molecular envelope using the algorithm of B.C. Wang. Acta Crystallogr A. 1987;A43:134–136.
[125] Karplus PA, Diederichs K. Assessing and maximising data quality in macromolecular crystallography. Curr Opin Struct Biol. 2015;34:60–68.
[126] Chen L, Oughtred R, Berman HM, et al. TargetDB: a target registration database for structural genomics projects. Bioinformatics. 2004;20:2860–2862.
[127] Slabinski L, Jaroszewski L, Rodrigues APC, et al. The challenge of protein structure determination – lessons from structural genomics. Protein Sci. 2007;16:2472–2482.
[128] Genest C, Weerahandi S, Zidek JV. Aggregating opinions through logarithmic pooling. Theory Decis. 1984;17:61–70.
[129] Yen S-J, Lee Y-S. Cluster-based under-sampling approaches for imbalance data distributions. Exp Syst Appl. 2009;36:5718–5727.
[130] Matthews BW. Comparison of the predicted and observed secondary structure of T4 phage lysozyme. BBA-Protein Struct M. 1975;405:442–451.
[131] Rao HB, Zhu F, Yang GB, et al. Update of PROFEAT: a web server for computing structural and physicochemical features of proteins and peptides from amino acid sequence. Nucleic Acids Res. 2011;39:W385–W390.
[132] Protlearn DT. Version 1. Posted: [cited 2020 Oct 3]. Available from: https://github.com/tadorfer/protlearn.
[133] LeCun Y, Bengio Y, Hinton G. Deep learning. Nature. 2015;521:436–444.
[134] LeCun Y, Boser B, Denker JS, et al. Backpropagation applied to handwritten zip code recognition. Neural Comput. 1989;1:541–551.
[135] Wilson J. Towards the automated evaluation of crystallization trials. Acta Crystallogr D. 2002;58:1907–1914.
[136] Marks DS, Colwell LJ, Sheridan R, et al. Protein 3D structure computed from evolutionary sequence variation. PLoS ONE. 2011;6:e28766.
[137] Agashe A, Mofrad MRK. Continuous distributed representation of biological sequences for deep proteomics and genomics. PLoS ONE. 2015;10:e0141287.
[138] Moult J, Pedersen JT, Judson R, et al. A large scale experiment to assess protein structure prediction methods. Proteins. 1995;23:ii–iv.
[139] Marks DS, Colwell LJ, Sheridan R, et al. Protein 3D structure predicted from evolutionary sequence variation. PLoS ONE. 2011;6:e28766.
[140] Asgari E, Mofrad MRK. Continuous distributed representation of biological sequences for deep proteomics and genomics. PLoS ONE. 2015;10:e0141287.
[141] Moult J, Pedersen JT, Judson R, et al. A large scale experiment to assess protein structure prediction methods. Proteins. 1995;23:ii–iv.
[142] Rohl CA, Strauss CEM, Misura KMS, et al. Protein structure prediction using rosetta. Methods Enzymol. 2004;383:66–93.
[143] Kabash W, Sander C. Dictionary of protein secondary structure: pattern recognition of hydrogen-bonded and geometrical features. Biopolymers. 1983;22:2577–2637.
[144] Liu DC, Nocedal J. On the limited memory BFGS method for large scale optimization. Math Program. 1989;45:503–528.
[145] Bahdanau D, Cho K, Bengio Y. Neural machine translation by jointly learning to align and translate. Posted: [cited 2014 Dec 1] Available from: https://arxiv.org/pdf/1409.0473.pdf.
[146] Devlin J, Chang M-W, Lee K, et al. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. Posted: October 11, 2018. Available from: https://arxiv.org/pdf/1810.04805.pdf.
Subject Index

(Artificial) neural network, (A)NN 58, 60, 61, 62, 66, 75, 78, 79, 86
Backpropagation 60, 76
Bayesian 63, 66
Boosting 61, 65, 85
(Class/sample) distribution 60, 67, 68, 69, 70, 74
Classification 60, 62, 63, 65
Confusion matrix 68, 72
Critical Assessment of protein Structure Prediction, CASP 57, 91, 92
Cryo electron microscopy, cryo-EM 56, 66, 92
Deep learners/learning 55, 57, 58, 66
Dimensionality 63, 67, 70
Exploratory data analysis, EDA 63, 67
Feedforward 60
Im/balance(d data) 70, 76, 81, 82, 85, 87, 88
Macromolecular (X-ray) crystallography, MX 55, 57, 66, 93
(Multi-layer) perceptron 66, 78
Performance 60, 63, 68, 72, 73, 74, 96
Receiver-operating characteristic, ROC 73
Regression 62, 63, 65, 66, 76, 80
Shallow learners/learning 65, 66
Un/supervised learning 62, 63, 72