Chapter 16

Deep Learning for Protein–Protein Interaction Site Prediction

Arian R. Jamasb, Ben Day, Cătălina Cangea, Pietro Liò, and Tom L. Blundell

Abstract

Protein–protein interactions (PPIs) are central to cellular functions. Experimental methods for predicting PPIs are well developed but are time and resource expensive and suffer from high false-positive error rates at scale. Computational prediction of PPIs is highly desirable for a mechanistic understanding of cellular processes and offers the potential to identify highly selective drug targets. In this chapter, details of developing a deep learning approach to predicting which residues in a protein are involved in forming a PPI—a task known as PPI site prediction—are outlined. The key decisions to be made in defining a supervised machine learning project in this domain are here highlighted. Alternative training regimes for deep learning models to address shortcomings in existing approaches and provide starting points for further research are discussed. This chapter is written to serve as a companion to developing deep learning approaches to protein–protein interaction site prediction, and an introduction to developing geometric deep learning projects operating on protein structure graphs.

Key words Protein, Structure, Protein–protein interaction, Deep learning, Structural biology, Machine learning, Graph, Geometric deep learning

1 Introduction

Proteins adopt complex three-dimensional structures in order to carry out cellular functions. Many of these functions are carried out by larger assemblies of protein complexes and regulated through physical contacts between effectors and regulators. Understanding protein–protein interactions (PPIs) is fundamental to understanding cellular processes in healthy and diseased states, and their accurate prediction is a longstanding goal of computational biology. Predicting the interacting residues involved in PPIs is useful for constructing refined PPI networks, understanding the impact of

The original version of this chapter was revised. The correction to this chapter is available at https://doi.org/10.1007/978-1-0716-1641-3_19

Daniela Cecconi (ed.), *Proteomics Data Analysis*, Methods in Molecular Biology, vol. 2361, https://doi.org/10.1007/978-1-0716-1641-3_16, © The Author(s) 2021, Corrected Publication 2021

mutations, improved accuracy in protein-protein docking and richer annotation of protein function [1]. Furthermore, predicting PPIs is desirable for structure-based drug discovery; PPI interfaces offer the potential for highly selective modulation of pathological processes [2].

Experimental methods for characterizing protein–protein interactions include Yeast-2-hybrid (Y2H) methods [3], mass spectrometry [4], tandem affinity purification [5], and protein chips [6]. However, these methods are time and resource intensive [7], and high false-positive rates are prevalent in larger experimental screens [8] limiting the scalability of protein–protein interaction characterization to the proteome level. There exist high-quality structurally annotated databases characterizing PPI sites such as BioLIP [9] (https://zhanglab.ccmb.med.umich.edu/BioLiP/), which collates structural interaction sites for a variety of protein interaction types. However, these databases characterize only a subset of extant proteins, and a significant proportion of the space remains unannotated and not structurally characterized. Furthermore, accurate predictions are made increasingly challenging due to the promiscuity of protein interactors; a given protein may have multiple interaction partners over disparate or overlapping regions of its surface. Proteins with multiple binding partners may interact with their partners at different times or feature large interaction sites capable of interacting with multiple partners simultaneously. Reporting of interactions may be biased; it has been shown that the number of reported interactions for a given protein is correlated with its frequency of occurrence in the literature [10]. Efficient and reliable computational prediction of protein–protein interactions is therefore highly desirable, though challenging.

Existing methods for PPI site prediction broadly fall into three categories. Protein-protein docking methods seek to produce structures of the resulting protein complex, and typically produce a number of scored candidate structural models as output [11]. Structure-based methods seek to perform prediction of interaction sites by leveraging protein structural information [12]. Sequence-based methods perform predictions based on protein sequences and form the bulk of the existing body of work due to the relative abundance of protein sequence data. While docking and structure-based methods typically require structural data, sequence-based approaches benefit from greater availability of data. However, structural methods may be limited in their utility for applications involving intrinsically disordered proteins (IDPs) or regions (IDRs), which play important roles in facilitating some PPIs, often allowing concerted folding and binding of sequences with these regions, and are enriched in protein and nucleic acid binding proteins [13]. The difficulty in structurally elucidating IDPs and IDRs means that structural datasets are typically deficient in disorder-mediated interactions. It should be noted, however,

that supervised sequence-based methods typically rely on datasets curated from structurally characterized protein–protein interactions (*see* Subheading 2.2 for details), where residues for which the solvent-accessible area decreases upon binding are considered interacting residues. Many machine learning-based approaches have the advantage that the bulk of the computational cost is incurred during training; inference from a trained model is relatively computationally inexpensive, whereas docking-based methods require large quantities of computational resources for each prediction to score and rank structures in the conformational space. An overview focused on machine learning and deep learning approaches and the requisite data preparation is provided in Fig. 1.

An example of a structure-based approach is ProtCHOIR (https://github.com/monteirotorres/ProtCHOIR), a tool for proteome-scale generation of homo-oligomers in an automated fashion, providing detailed information for the input protein and output complex (Torres PHM & Blundell TL, Manuscript in preparation). ProtCHOIR requires input of either a sequence or a protomeric structure that is queried against a pre-constructed local database of homo-oligomeric structures, then extensively analyzed using well-established tools such as PSI-Blast [14] (https://blast.ncbi.nlm.nih.gov/Blast.cgi?PAGE_TYPE=BlastDocs&DOC_TYPE=Download), MAFFT [15] (https://mafft.cbrc.jp/alignment/software/), TMHMM [16] (http://www.cbs.dtu.dk/services/TMHMM/), PISA [17] (https://www.ccp4.ac.uk/MG/ccp4mg_help/pisa.html), Gesamt [18] (http://ccp4serv7.rc-harwell.ac.uk/gesamt/), and Molprobity [19] (http://molprobity.biochem.duke.edu). Finally, MODELLER [20] (https://salilab.org/modeller/) is employed to achieve the construction of the homo-oligomers. The output complex is thoroughly analyzed taking into account its stereochemical quality, interfacial stabilities, hydrophobicity and conservation profile. The software is easily parallelizable and also outputs a comma-separated value file with summary statistics that can straightforwardly be concatenated as a spreadsheet-like document for large-scale data analysis.

This chapter focuses on the considerations involved in applying deep learning methods to protein structure data for the prediction of protein–protein interaction sites. The main steps in developing such a project, from data collection and preparation, featurization and representation, through to model design and evaluation are highlighted. The choice of representation is a key decision in such an undertaking. There exist in the literature many machine learning-based approaches to this problem covering a range of classical models and representations including: logistic regression [21], Naive Bayes classifiers [22], Support Vector Machines (SVM) [23–25], and random forests [26–28]. A full review of existing approaches is beyond the scope of this chapter. Neural network-

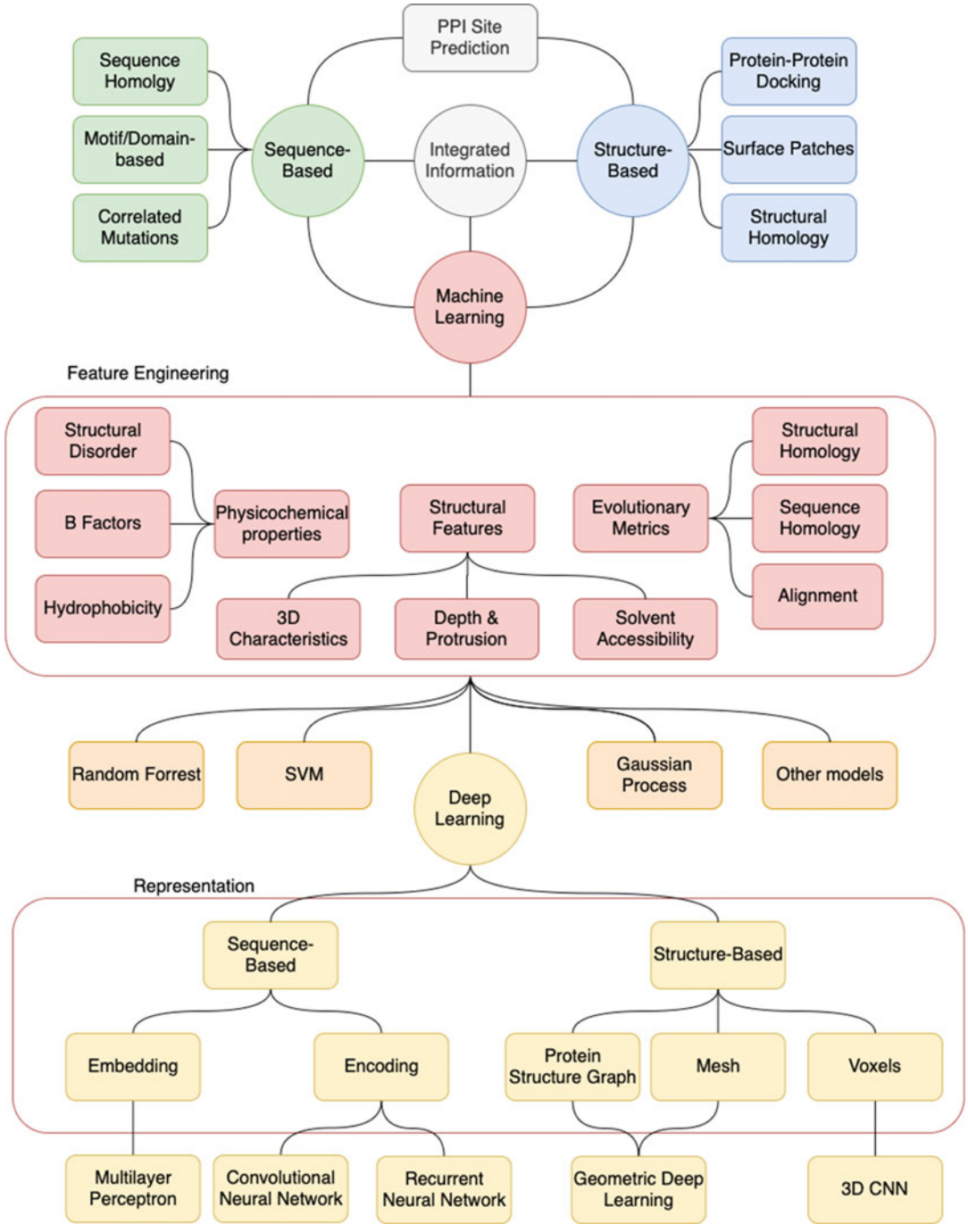

Fig. 1 Overview of machine learning and deep learning approaches to protein–protein interaction site prediction. Input structural or sequence data requires feature engineering or transformation into an appropriate representation for the architecture of the model

based approaches have included shallow neural network models with limited width (models with comparatively few hidden layers and limited dimensionality compared to more recent architectures) [29–32] and, more recently, deeper architectures. Larger datasets and GPU acceleration have enabled the training of deeper neural network architectures. Amongst the more recent deep learning approaches, convolutional neural networks (CNNs) [33], recurrent neural networks (RNNs) including Long Short-Term Memory networks (LSTMs) [34] and hybrids thereof [35] have been applied to the sequence-based interaction site prediction problem. Here the authors note a novel problem framing where models operate directly on graph-structured representations of protein structures (*see* **Note 1**) that has recently developed using Message Passing Neural Processes [36]. The graphs are composed of constituent amino acid residues and their interactions, and the target labels are binary labels indicating whether or not a particular residue takes part in a protein–protein interaction. The authors believe this is timely due to rapid development and early successes in geometric deep learning and its applications to computational structural biology [37–42], and indeed applications to the study of protein–protein interactions [37]. Indeed, the work of Fout et al. [38] where the authors applied a Graph Neural Network (GNN) model to PPI interface prediction between two interacting proteins is acknowledged.

There are a number of problems to be aware of in designing machine learning predictors. For instance, data often suffer from class imbalances, where the number of negative sites (non-interacting residues) is much greater than the number of positive sites (*see* **Note 2**). This problem is exacerbated in larger proteins as the fraction of positive sites decreases with size [29, 43] and has been shown to bias predictors that do not account for this [44]. Sequence-based predictors have also been shown to confuse small molecule ligand, DNA, and RNA binding regions with protein-binding regions [45].

2 Materials

2.1 Computing Resources

Most of the development workflow can be performed on a standard UNIX workstation equipped with a GPU suitable for training deep learning models. The exact GPU memory requirements will depend on the model architecture and dataset sizes used. If models are to be trained on local or cluster-based GPU acceleration (strongly recommended for training and running models at scale), an installation of an appropriate CUDA (https://developer.nvidia.com/cuda-toolkit) version compatible with the GPU will be required. The user should be familiar and comfortable with running command line tools, base python, a commonly used deep

learning framework (such as PyTorch or Tensorflow) as well as installing python packages and machine learning basics.

2.1.1 Software Installations

It is recommended to install the required packages in a virtual environment. A virtual environment can be set up using Conda [46] (https://docs.conda.io/en/latest/), a commonly used package and environment manager.

2.1.2 Machine Learning Frameworks

There are a number of actively developed machine learning frameworks. A popular choice for traditional ML is SciKit-Learn [47] (https://scikit-learn.org/stable/). For deep learning frameworks, popular choices include: PyTorch [48] (https://pytorch.org), TensorFlow [49] (https://www.tensorflow.org), and Theano [50] (http://deeplearning.net/software/theano/). Each framework has an associated ecosystem of community-developed implementations of popular methods and tools. A fuller discussion of popular frameworks can be found in a review by Erickson et al. [51].

2.2 Databases and Datasets

There are a number of databases which collect relevant data for constructing datasets of protein–protein interactions. Zhang et al. [52] collated a large database of protein sequences, annotated with protein, small molecule ligand and nucleic acid binding sites at the residue level. Li et al. used this resource to create a large training dataset by removing training data sequences with >40% sequence similarity to the test datasets and >40% to other training examples to ensure diversity in the training set [35]. The authors of this work make available training (87.5%, $n = 9681$) and validation (12.5%, $n = 1382$) splits. This is, to our knowledge, by far the largest dataset collated for PPI site prediction.

In addition, there are four other processed datasets that can be used for training and testing: Dset_72 [53], Dset_164 [54], Dset_186 [22], and Dset_448 [21], where the tail number indicates the number of sequences in each dataset. Dset_72, Dset_164, and Dset_186 are constructed through curating heterodimeric PDB entries with structural resolution <3 Å and <25% sequence identity. It is suggested that these datasets be used as independent test sets (practitioners should take care to avoid overlap with these datasets in their training data, discussed in Subheading 3.1.2) to enable benchmarking of novel methods against existing approaches.

2.3 Tools for Computing Features and Representations

There exist many tools for computing both sequence-based and structure-based features for proteins. Previous research by Jones and Thornton [55] into identifying important features for PPI site prediction has revealed solvation potential, residue interface propensity, hydrophobicity, planarity, protrusion, and accessible surface area (ASA) as important features to discriminate between binding residues. In the intervening years, many more tools for calculating protein and amino acids properties from sequences and

structures have been made available. Example tools and featurization options that practitioners may wish to consider in developing a PPI site prediction project are outlined.

2.3.1 Sequence-Based

Sequence-based tools include: PROFEAT [56, 57] (http://bidd.group/cgi-bin/profeat2016/ligand/profnew.cgi), a longstanding web server for calculating structural and physicochemical properties for protein sequences. ProPy [58] (https://pypi.org/project/propy3/1.0.0a2/) is a python package capable of calculating a large number of structural features from protein sequences (amino acid composition descriptors, dipeptide composition descriptors, tri-peptide composition descriptors, Normalized Moreau-Broto autocorrelation descriptors, Moran autocorrelation descriptors, Geary autocorrelation descriptors, Composition, Transition, Distribution descriptors (CTD), sequence order coupling numbers, quasi-sequence order descriptors, pseudo amino acid composition descriptors, amphiphilic pseudo amino acid composition descriptors). Putative Relative Solvent Accessibility (RSA) can be computed using ASAquick [59] (http://mamiris.com/ASAquick/). Meiler et al. [60] make available a set of low-dimensional embeddings of the physicochemical properties of amino acids that can be used for featurization. Position-specific scoring matrices (PSSMs) are often highly informative features [33], as residues important for facilitating interactions are likely to be evolutionarily conserved and can be readily computed using PSI-BLAST [14]. Similarly, evolutionary conservation (ECO) can be computed from HHBlits [61, 62] (https://github.com/soedinglab/hh-suite). Further to these descriptors, Li et al. [35] make use of several other descriptors including: High-Scoring Pairs (HSP), which are similar sub-sequences between two proteins scored using scoring matrices such as PAM & BLOSUM [63] using SPRINT [64] (https://github.com/lucian-ilie/SPRINT/). ANCHOR [65] (https://iupred2a.elte.hu) can be used to calculate the putative protein-binding disorder. Hydrophobicity information can be encoded using the hydropathy index computed using [66].

2.3.2 Sequence Embeddings

Embedding-based methods take sequences as input and return a fixed-length representation of the sequence. The goal of sequence embeddings is to keep similar sequences close in the embedding space, while maintaining distance between dissimilar sequences. There are a number of pre-trained sequence embedding models that can be used with protein sequences. For instance, ProtVec [67] is one such method that has shown good results when applied to protein family classification. ProtVec has been leveraged in PPI site prediction in work by Li et al. [35] in which the authors summed the 100-dimensional representations of each of the sequence 3-mers to produce the fixed embedding and speed up computation. UniRep [68] provides a pre-trained RNN model trained on

24 million sequences from UniRef50 [69]. Rives et al. [70] make available a pre-trained transformer model trained on over 250 million sequences that can be utilized to generate sequence embeddings for PPI tasks.

2.3.3 Structure-Based

DSSP [71, 72] (https://swift.cmbi.umcn.nl/gv/dssp/DSSP_3.html) is a longstanding tool for calculating secondary structural descriptors of proteins from their structures. Graphein [73] (https://github.com/a-r-j/graphein) is a python library for computing geometric representations of protein structures. It is capable of flexibly creating protein structure graphs at various levels of granularity (amino acid, atomic) and under various construction schemes (based on intramolecular contacts and/or various distance-based methods) and mesh representations of protein surfaces. It also includes various featurization schemes, such as the aforementioned DSSP descriptors, cartesian coordinates, and the low-dimensional embeddings of the physicochemical properties of amino acids. Voxelized representations of 3D atomic structures where atoms are fixed as points on a 3D grid can be featurized using a variety of atomic descriptors such as: encodings of atom-type, atomic number, atomic mass, explicit and implicit valence, hybridization, ring status, aromaticity, formal charge, and chiral status as additional channels.

3 Methods

3.1 Data

Training and evaluating a supervised deep learning model requires datasets of labeled examples. These data should be partitioned into training, validation, and test datasets. The raw data should be converted into an appropriate representation and pre-processed

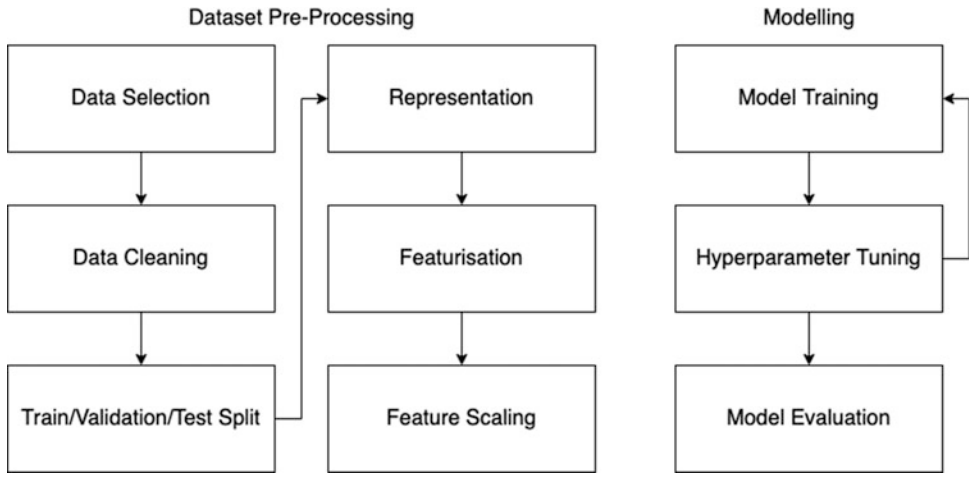

Fig. 2 Overview of data processing and model development pipeline

prior to training and inference. Important commonalities and considerations in preparing a dataset are highlighted in this section. A schematic overview of a standard preparation procedure is shown in Fig. 2.

3.1.1 Curation

First, a set of protein structures or sequences with protein–protein interaction binding site annotations is required. Some of the available processed and raw sources for such data are discussed in Subheading 2.2.

3.1.2 Train-Test Split Strategies

Data should be split into training, validation, and testing data. Training data are used to iteratively train the parameters of the model while validation data and testing data are used to evaluate model performance for hyperparameter selection and final evaluation, respectively. The proportions 80/10/10 are commonly used for training/validation/testing data. The experimenter can make a number of choices with respect to splitting data. The most obvious and straightforward solution is to split the data randomly. However, this can produce misleading results if related sequences are over-represented in the data compared to true protein space. Furthermore, highly similar training and testing examples may reward predictions based on homology, rather than learning the data-generating function. Both of these scenarios violate the independent and identically distributed (i.i.d.) assumption, where one assumes that the training and test data are drawn from the same underlying joint distribution. In developing a model, one is interested in performing predictions on novel protein structures that may lie outside of this distribution (ideally a dataset with a representative distribution is constructed), and so a practitioner wishes to examine the generalization of the model, i.e., the extent to which the model can make accurate predictions on unseen data, indicating learning of the underlying physical process, rather than memorization of the input data points. Thus, a good test set should contain a set of protein structures distinct from the structures the model was trained on. However, the exact manner by which one can achieve this depends on the application domain. For instance, if the experimenter is looking to build a model that is specific to predicting interaction sites on, say, kinases, then a random partitioning of the data could be considered valid given a sufficiently large and diverse dataset of annotated kinase structures.

While often used, splitting should not be performed on the basis of clustering by sequence similarity (e.g., using tools such as BLAST [74, 75]) or thresholding by sequence similarity. Instead, the experimenter should make use of resources such as SCOP [76] (http://scop.mrc-lmb.cam.ac.uk) or CATH [77] (https://www.cathdb.info) to obtain information about related structures in order to prevent leaking data from the training set into the test set in the form of structural homologs. This is important as

sequences with 35–40% sequence similarity are highly likely to adopt similar structural folds [78, 79]. Furthermore, structural similarity has been found between proteins with sequence similarity in the 8–30% range [80]. However, many PPIs are mediated at the domain level, and similarity at this level is not captured by simplistic sequence similarity scores. These can be thought of as structurally and somewhat evolutionarily independent subunits. Thus, more nuanced data partitioning based on knowledge-based annotation databases should be standard practice in developing a well-principled project. It should be noted that this presents a much more challenging, but more representative, task for a classifier and obtaining good predictive performance will be difficult but more widely applicable.

3.1.3 Representation

Protein structures are not data structures and there is a large amount of flexibility available to the practitioner in deciding how to represent them. Some common choices are outlined in Table 1. As discussed previously, the choice of representation is central to designing a machine learning project. It informs the choice of architecture and techniques available to the experimenter.

Table 1
Summary of different choices of representations for protein structures

Representation	Model	Pros	Cons
Sequence	RNN LSTM CNN GRU	Large quantities of protein sequence data available Handles inputs of varying sizes Pre-trained sequence-based language models for producing embeddings	Poor structural signal
Grid	3DCNN	Have shown strong performance on structural tasks [81–85]	Do not effectively handle inputs of varying sizes Not rotationally invariant Computationally expensive. Studies typically limit the inputs to voxels containing only the area of interest Require structural data
Graph	GNN	Representation captures the internal chemistry of the protein Handles inputs of varying sizes Large number of possible construction schemes	Require structural data

A good representation will infuse the model with a strong *inductive bias*. An inductive bias allows a model to prioritize one solution to a problem over another and can express assumptions about the underlying data-generating process or the solution space [86]. For instance, it makes sense to perform 1D convolutions over protein sequences and 3D convolutions over grid-structured representations of protein structures as the spatial components of the convolutions are meaningful and valid. Meaningful inductive biases in the case of PPI site prediction may include the ability to account for both long- and short-range interactions between constituent amino acids. For instance, residues that are distant from one another in the sequence may be found in the same interaction interface.

The choice of representation also affects the applicability of the trained model. For instance, a model trained using a sequence-based representation will be able to perform predictions on unseen sequences. However, a model trained on a representation derived from crystal structures will require crystal structures in order to make predictions after training (learning using privileged information offers a framework in which both sequences and structures can be leveraged during training while allowing for sequence-based predictions during inference. *See* Subheading 4). The desired use case is therefore an important consideration in the choice of representation and should be balanced with the availability of data and relevant inductive biases in the decision-making process.

3.1.4 Input Features

Featurization of the input data can be performed using various stand-alone programmatic bioinformatics tools and web servers according to the representation selected by the experimenter. A non-exhaustive collection of these is highlighted in Subheading 2.3.

3.1.5 Pre-processing

Data and labels used in a machine learning model should be scaled to enable efficient training (*see* **Note 3**). Training features and real-valued labels should be pre-processed using a standard scaler (i.e., zero mean and unit variance). This training scaling transformation should be kept and applied to any validation or test data used. It is important to note that the scaling is performed on the training data, and this same transformation is applied to the validation and test sets, rather than scaling each dataset independently. This step is performed to ensure that different features with different scales are standardized to prevent the accumulation of large spread of weights in the network. Large weights are often unstable, as they may produce large gradient values, causing large updates to the network, or result in dead neurons in the network (*see* **Note 8**). Alternative pre-processing techniques for numeric features can include min-max scaling and transformations, such as

log-transforming features that range across many orders of magnitude. Categorical and ordinal features should be numerically encoded, e.g., through one-hot encoding.

3.2 Model Evaluation

The goal during training is to minimize a loss function by iteratively updating the model parameters (*see* **Note 4**). During each training epoch, the model is presented with *mini-batches,* subsets of the training data, in between which weights are updated (*see* **Note 5**).

3.2.1 Hyperparameter Tuning

Deep learning models are themselves described by *hyperparameters*, which describe the various aspects of the model architecture and the training regime. The set of all possible hyperparameters forms a space, from which the objective is to approximate the combination of hyperparameters that lead to the best predictive performance at test time (*see* **Note 6**). A selection of common hyperparameters and commonly used values is described in Table 2. It should be noted that different architectures have additional hyperparameters associated with them. For instance, CNN-based models should include convolution filter sizes, padding, stride length, and pooling types; graph-based methods will have hyperparameters such as the aggregation and readout functions; sequence-based architectures have hyperparameters such as the embedding dimension. Furthermore, some hyperparameter choices have additional hyperparameters associated with them, such as the momentum term used in Adam optimization.

3.2.2 Evaluation Metrics

In order to determine the optimal set of hyperparameters, practitioners are required to make comparisons between models and their performance. When is a model considered better than another? What is considered to be the key measure of performance? These questions fall into the categories of model assessment and selection.

Ideally, the true error of the classifier is approximated. Broadly speaking, there are two methods for approximating this and evaluating the performance of a model: Cross-Validation (CV) based methods and using training/validation/testing data splits. Within CV, there are two main paradigms to consider. The first is k-fold CV, where the data are split into k segments and train on $k - 1$ of the folds, assessing the performance on the left-out fold. This is performed k times and the errors averaged, to assess the model. The second paradigm is leave-one-out-cross-validation (LOOCV), which is a special case of K-fold CV, where $K = n$ (the total number of examples in the dataset.). Using CV-based methods for evaluating deep learning models can be difficult in practice, due to the high computational and time requirements associated with model training. Training/Validation/Test data splits are useful compromises, where the data are partitioned into three sets. The model is trained on the training data, the hyperparameters are tuned using the performance on the validation data as a guide, and the final

Table 2
Common hyperparameters found across neural network architectures with commonly used choices or ranges

Hyperparameter	Type	Scope	Description
Architecture			
Loss function	Categorical	{Cross entropy, Hinge, MSE Ordinal}	The loss function measures the prediction error associated with the current state of the model. This function is minimized by the optimizer during training
Layer width	Integer	0 <. Typically [8–1024]	The number of hidden units in a given layer
Activation functions	Categorical	{Sigmoid, tanh, ReLU}	Non-linear functions applied to the output of hidden units
Model depth	Integer	[1–100]	The number of layers in the model
Weight initialization	Categorical	{Zero, Random, Xavier [87]}	Controls the starting values of the model weights
Dropout	Float	$0 < d < 1$. In practice, {0, 0.25, 0.5, 0.75} are common choices	Dropout is a regularization technique where some hidden units in a layer are "switched off" with some probability during each training epoch
Regularization	Categorical	{L1, L2}	Regularization adds a penalty to the loss function that penalizes large weights
Training			
Learning rate	Float	[1e-5–1]	Controls how strongly a network updates its weights by scaling the error gradients. *See* **Note 4**
Learning rate schedule	Categorical	Constant, cyclic, decay	Controls adjustment of the learning rate throughout training
Batch size	Integer	[8, 16, 32, 64, 128, 256]	The number of training examples presented to the network between each weight update
Batch normalization	Boolean	{True, False}	Controls whether each training batch is normalized
Optimizer	Categorical	SGD, ADAM, rmsprop	Optimizers update the network weights to minimize the loss function
Epochs	Integer	[0–1000]	The number of times the training dataset is presented to the network during training

Some hyperparameter choices will have additional hyperparameters associated with them. This table is not exhaustive and does not include architecture-specific hyperparameters

assessment of the model is carried out on the test data. It is important to prevent leakage of information between the sets (e.g., having very similar training and testing examples, discussed in Subheading 3.1.2) as this prevents accurate model assessment.

Table 3
Confusion matrix displaying possible classification outcomes

		True labels	
		Y	**N**
Predicted labels	Y	True positive (TP)	False positive (FP)
	N	False negative (FN)	True negative (TN)

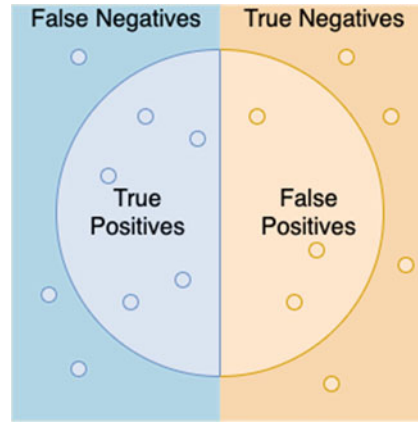

Fig. 3 Illustration of classification error types

Furthermore, hyperparameters should only be tuned on the basis of training and validation performance; hyperparameter tuning on the test set performance is another form of leakage.

To understand and evaluate model performance, a selection of classification metrics should be examined. To begin, consider the possible outcomes of a prediction with respect to its true label. Table 3 is known as a *confusion matrix* and the relevant metrics discussed below are dependent on the desire types of classification correctness. It is important to note that PPI site prediction is an *imbalanced* classification problem: positive and negative labels do not occur at similar rates in the data (*see* **Note 2**). Thus, simple metrics, like the fraction of correct predictions, are inappropriate as the null predictor (a model that predicts 0 for each output) would score according to the frequency of inactive residues in the sequence data. Thus, a rigorous evaluation would leverage an ensemble of metrics to assess a model and understand its performance on each of prediction classes outlined in the confusion matrix. These are presented in the definitions below (Fig. 3). Readers should note there are a number of ways of computing these scores, e.g., on a per-residue or per-protein basis (*see* **Note 7**).

Sensitivity *This is equivalent to the fraction of correctly identified interaction sites. This is also referred to as recall or the true positive rate (TPR).*

$$\text{Sensitivity} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

False-Positive Rate (FPR) *This is the fraction of negative sites wrongly classified as positive over the total number of negative sites.*

$$\text{FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}}$$

Specificity *This is equivalent to the fraction of correctly classified non-interacting sites out of all the non-interacting sites.*

$$\text{Specificity} = \frac{\text{TN}}{\text{TN} + \text{FP}}$$

Precision *This is equivalent to the fraction of correctly classified interactions sites out of all interaction sites.*

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

Accuracy *This is equivalent to the fraction of correctly predicted interacting and non-interacting sites.*

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FN} + \text{TN} + \text{FP}}$$

ROC and AUC *The Receiver-Operator Characteristic curve (ROC) plots the TPR vs FPR at different classification thresholds. The Area Under the ROC Curve (AUC) gives a measure of performance across classification thresholds from 0 to 1.*

F1 Score *This is the harmonic mean of the precision and the sensitivity and ranges between 0 and 1, where one indicates perfect precision and recall.*

$$F1 = 2 \times \frac{\text{Sensitivity} \times \text{Precision}}{\text{Sensitivity} + \text{Precision}}$$

Matthew's Correlation Coefficient (MCC) *This measure ranges between −1 and 1. A high score is only achievable if a binary classifier performs well in all the metrics in the confusion matrix. It has the advantage of being robust to class-imbalanced datasets and provides the most truthful measure of classifier performance [88]. Therefore, obtaining a strong score with this metric will be the most challenging of the metrics outlined here, and should form the basis for assessing the performance of a protein–protein interaction site-prediction classifier.*

$$\text{MCC} = \frac{\text{TP} \times \text{TN} - \text{FN} \times \text{FP}}{\sqrt{(\text{TP} + \text{FN}) \times (\text{TP} + \text{FN}) \times (\text{TN} + \text{FP}) \times (\text{TN} + \text{FN})}}$$

3.2.3 Overfitting

In the development process, it is important to report metrics for both the training and validation datasets in order to identify overfitting. Overfitting arises from a combination of data scarcity and a model that is too flexible, resulting in "memorization" of the training data, rather than learning of the underlying process. Techniques for countering overfitting include adding regularization penalties to the loss function, using dropout or reducing the capacity of the model (e.g., restricting depth or the size of the hidden layers). *See* **Note 8** for more details regarding regularization techniques.

3.2.4 Attribution

Most machine learning and deep learning methods are black-box predictors that do not allow for a clear and interpretable examination of the relationship between the input features and the output classification of the predictor. However, there are techniques that allow for some exploration of this relationship, such as gradient attribution methods [89] and attention mechanisms (Subheading 4) through visualization of the attention weights.

3.3 Alternative Training Regimes for Future Model Development

Deep learning has demonstrated excellent performance in a variety of tasks within computational biology, and indeed computational structural biology. Supervised deep learning involves training with data an artificial neural network that learns to perform regression or classification tasks. The training process involves iteratively tuning the weights of a series of feedforward feature-extracting layers. These layers learn hierarchical features based on their inputs, such that deeper layers in the network learn higher-order features as a composition of earlier layers. Such methods have already been applied successfully to PPI site prediction. While this has resulted in impressive performance gains, PPI site prediction is far from a solved problem and the utility of a highly-performant predictor demands further development in this area. This requires improvements in both training and benchmark dataset quality, as well as in the modeling process. In this section, alternate training regimes that may provide utility for researchers interested in furthering future model development in this field are highlighted.

3.3.1 Multi-modal Input

Combining additional data modalities has proven useful in PPI site prediction. For instance, Zeng et al. [33] make use of a combination of local and global sequence based features for their predictor. Local features are captured using a sliding window over the amino acid sequence, and global features through the use of a text CNN, and their contributions validated through feature-ablation studies. When utilizing multiple data modalities, the question arises as to whether these features should be fused early or late in the model architecture and whether they require individual input encoders to extract features. Furthermore, the correspondence of the modalities should be taken into account, e.g., is each training example

presented as a tuple x_i, x_j or is the second modality repeated across certain examples. Feature ablation studies are useful to understand the relevant contributions of each modality (*see* **Note 9**).

3.3.2 Transfer Learning

Transfer learning involves training a model on a similar task, which can then be fine-tuned to the primary task. Such fine tuning typically involves replacing or adding to the latter layers of the model and re-training on the primary dataset. The intuition here is that the earlier layers of the model have learned useful representations relevant to the task and can thus help improve performance and speed up convergence. Practitioners using such a strategy may consider whether or not to freeze the weights of the pre-trained layers during the fine-tuning procedure and conceive of the pre-trained layers as fixed feature extractors.

3.3.3 Multi-task Learning

Multi-task learning (MTL) involves training a predictor on multiple related tasks simultaneously, typically optimizing using additional auxiliary losses with the goal of improving predictive performance or generalization. The intuition underlying this is that the related tasks contain related training signals that are useful to the model to better learn the main task through inductive transfer. For instance, in the case of PPI site prediction, Zhang et al. make use of a multi-task framework to jointly predict interaction sites and solvent-accessible residues as only solvent-accessible residues are capable of interacting with another protein and identify this as an effective strategy in countering the class-imbalance problem [34]. Further model development leveraging multi-task learning may like to examine joint prediction of PPI sites and nucleic acid or small molecule binding sites, an approach developed by [21] to address cross-prediction of these sites.

3.3.4 Learning Using Privileged Information

Learning Using Privileged Information (LUPI) is an approach where a model is trained on multiple sources of data, but evaluated and deployed on examples where some of these data sources are unavailable [90]. Formally, training examples are presented as triples $\left(x_i, x_i^*, y_i\right)$ (instead of tuples (x_i, y_i) in the classical case), where x_i^* is the additional information for training example x_i and y_i is the corresponding label. For instance, a LUPI approach to PPI site prediction could involve training a model on protein structures and sequences, where available, which is then evaluated on and deployed to predict interaction sites from sequences alone. Such an approach has been developed for sequence-based protein-ligand binding affinity prediction, to produce results comparable to structure-based approaches [91]. LUPI presents an attractive framework in which multiple data modalities can be used without requiring complete coverage for all the examples; this is especially relevant in the context of biological datasets which are often

fragmented in this respect as they are typically collected without a strategy amenable to machine learning projects in mind.

3.3.5 Uncertainty Modeling and Active Learning

Modeling of uncertainty is of vital importance in computational biology. In data-limited scenarios, as is often the case in computational structural biology, uncertainty modeling can avoid overconfident predictions and enable judicious exploration of space outside of the training distribution [92, 93]. Uncertainties are especially vital in experimental contexts where the acquisition of additional data points is slow, arduous or expensive, as they can allow the experimenter to prioritize hypotheses with a high likelihood of success or experiments of potential greater novelty and higher associated risk [94]. Furthermore, uncertainty modeling opens the possibility for active learning loops, where further exploratory and validation experiments can be prioritized using the model and an associated *acquisition function*, and the new data incorporated into the model iteratively to explore increasingly distant regions of biological space [95].

Acquisition functions are used to propose which examples in the search space should be considered next by a model. These functions are typically inexpensive to evaluate, and examples include greedy approaches, where data points with the highest predicted response are prioritized, variance-based, where data points with the highest-associated prediction variance are prioritized or expected improvement based methods. Other commonly used acquisition functions include Upper Confidence Bound (UCB) and Maximum Probability of Improvement (MPI).

3.3.6 Attention Mechanisms

Attention mechanisms allow models to apply learnable weights to subsets of their inputs, depending on their perceived importance. Co-attention is a mechanism useful in multi-input models, which allows the attention on each input to be *conditional* on the other input. For instance, when designing a model to predict PPI sites *between* two proteins, co-attention mechanisms may allow the model to attend to more relevant parts of each protein in a manner specific to that particular interaction. When presenting one of these proteins with a different interaction partner, the model may attend to different parts of the proteins if the interaction occurs in a different region. Attention mechanisms have successfully been applied to paratope prediction [96].

3.3.7 Ensembling

Ensemble models involve constructing multiple models to perform predictions on the same input. The final prediction then results from some form of averaging of all of the individual model predictions. This can take the form of voting (e.g., as in Random Forests), averaging or by training another model that takes these predictions as inputs. It is often recommended to ensemble models with different architectures as each will have different biases and therefore

make different mistakes, making the ensemble model more robust and more beneficial. It is possible to ensemble models of the same architecture, e.g., through snapshot ensembling [97], where "snapshots" of the weights are taken throughout the training of a single model and ensembled to create the final predictor. These approaches can be thought of as ensembling in model space, as several models are constructed and ensembled. This can be computationally costly, due to the requirements of constructing multiple models during training and performing several predictions during inference. Ensembling methods have previously been applied to PPI site prediction [98]. Weight averaging is another ensembling technique that can be considered ensembling in weight space as the procedure results in a singular model.

4 Notes

1. Graphs, $G = (V, E, X^v, X^e)$, are structures used to model objects consisting of a set of nodes (or vertices), $v_i \in V$. with, typically, pairwise relations, $e_{i,j} = (v_i, v_j) \in E$, between them such that $E \subseteq V \times V$. Node features, $X^v \in \mathbb{R}^{|V| \times d}$; $x_i^v \in \mathbb{R}^d$, are the d features associated with node v_i. Similarly, edge features, $X^e \in \mathbb{R}^{|E| \times c}$ and $x_{i,j}^e = x_{v_i,v_j}^e \in \mathbb{R}^c$ are the c features associated with edge x_{v_i,v_j}^e. Protein structures can be represented as graphs of residues joined by intramolecular interactions or some distance-based criteria, such as thresholding of Euclidean distance or on the basis of K nearest neighbor clustering of distances. Protein structure graphs have been used in computational biology for prediction of protein–protein interaction interfaces [38] and protein structural classification [41]. One can distinguish between geometric deep learning methods, which operate directly on the graph structure, and machine learning applied to graph-based features. For instance, graph-based signatures have proved effective in a variety of applications relating to the impact of mutations on protein interactions [99, 100]. However, recent developments in graph representation learning offer the potential to exploit the relational structure of the data as an inductive bias in the model.

   In this problem-setting, PPI site prediction becomes a *node classification* task, where the objective is to predict a binary label, $\hat{y} \in \{0, 1\}$, for each amino acid in the structure graph, indicating whether or not it partakes in a protein–protein interaction. In order to do this, graphs are enriched with information about the protein structure in the form of node and edge features. These features provide the basis for computing the message passing signals in the model to exploit the

relational structure present in the data. Node features can take the form of an encoding of the amino acid residue type, secondary structure information, surface accessibility metrics, cartesian coordinates of centroid position and PSSMs. Additional features highlighted in Subheading 2.3 can be computed and used to enhance the modeling by practitioners. Edge features can be used to specify the intramolecular interaction type or Euclidean distance between two adjacent nodes. The authors have developed an example of this type of approach using Message Passing Neural Processes [36].

2. Protein–protein interaction data and site annotations are heavily class-imbalanced as negative results are not often reported or collated. Tools such as SMOTE [101], class weighting techniques, or loss functions that account for this can be used [34].
3. It is typically best to perform some normalization/scaling on input features. The intuition for this can be to ensure equal contribution from each predictor to the model. However, one should be careful when scaling quantities measured in the same units and whether it makes sense to normalize some features at all.
4. Learnable parameters in the model are iteratively tuned during training by backpropagating computed error gradients. Various optimization techniques for this procedure exist such as Stochastic Gradient Descent (SGD) and Adam [102]. The optimization technique is itself a hyperparameter of the architecture, and there may be additional hyperparameters associated with it, such as the learning rate. The learning rate is a small positive value (typically $[0, 1]$), and can be thought of controlling the "speed" at which a model learns by scaling the size of the weight updates in response to the computed error gradient. Large learning rates typically allow for fast learning, at the cost of potentially resulting in a sub-optimal set of weights. Smaller values will require more epochs to train but may improve the final set of weights by finding better quality minima. The learning rate is often considered to be the most important hyperparameter to tune [103]. Typical starting points are 0.001, 0.01, or 0.1. Learning rate schedulers can be used to vary the learning rate over training time. A fuller discussion of optimization techniques can be found in [104].
5. Training data should be shuffled between training epochs in order to reduce variance and reduce overfitting. Imagine a dataset where the examples are ordered by class. When a minibatch is selected from this dataset, it is desirable that the minibatch is representative of the true dataset in order to estimate the true error gradient. If the dataset is ordered in some manner, this is not achieved. In the context of regular SGD,

shuffling is beneficial as it ensures each example produces an independent change in the model that is not biased by ordering artifacts.

6. Deep learning models have many learnable parameters within the model, whereas the architecture as a whole is determined by hyperparameters. Non-exhaustively, these include the number of epochs, the size and depth of the layers, the size of each training batch, and the learning rate. These hyperparameters form a search space that must be traversed to produce a performant predictor. Various strategies for this include: random search [105], grid search and Bayesian optimization techniques [106].
7. When calculating evaluation metrics for predictions, the practitioner is left with the question of whether to compute the metrics for each protein and average them (micro-averaging) or to compute the metrics across all the individual amino acid predictions in the dataset (macro averaging). While macro results are typically reported in the literature, it is recommended to monitor both during development in order to gain a fuller understanding of the model. For instance, this will allow for better understanding the effect of protein size on performance.
8. Overfitting occurs when the model too closely fits the training data and generalizes to unseen data points poorly. This can be observed by comparing the performance on the training and validation datasets. A variety of techniques is available for combatting overfitting. Early stopping involves monitoring the validation loss during hyperparameter tuning and stopping the training process if the loss starts to increase. As this can fluctuate, a patience hyperparameter can be used where training is allowed to continue for a number of subsequent epochs and halted if no improvement is seen. Regularization is a family of techniques that includes adding penalties to the loss function based on the size of the model weights. Large weights can result in large updates to the network, a phenomenon known as *exploding gradients*, which can result in "dead" neurons, whereby they are shifted away from the training manifold. Dropout is another technique, where subsets of neurons are "switched off" randomly during each training epoch to encourage learning distributed internal representations and prevent over-reliance on individual neurons.
9. Feature ablation studies involve training a model on restricted subsets of the available features in order to interrogate their usefulness and contribution to the final model performance. This can also be used to perform feature selection. Due to the *curse of dimensionality* (where the dimensionality of the

predictors is greater than n, the number of training examples) which results in the training data space being very sparse, identifying highly predictive features is important to improve prediction accuracy and reduce overfitting to the training data.

Acknowledgments

ARJ is supported by a BBSRC DTP studentship. CC is funded by DREAM CDT. TLB thanks the Wellcome Trust for an Investigator Award (200814/Z/16/Z; 2016 -) for support of this research.

References

1. Zhang C, Freddolino PL, Zhang Y (2017) COFACTOR: improved protein function prediction by combining structure, sequence and protein–protein interaction information. Nucleic Acids Res 45(W1):W291–W299. https://doi.org/10.1093/nar/gkx366
2. Jubb H, Higueruelo AP, Winter A et al (2012) Structural biology and drug discovery for protein–protein interactions. Trends Pharmacol Sci 33(5):241–248. https://doi.org/10.1016/j.tips.2012.03.006
3. Ito T, Chiba T, Ozawa R et al (2001) A comprehensive two-hybrid analysis to explore the yeast protein interactome. Proc Natl Acad Sci U S A 98:4569–4574
4. Gavin A-C, Bösche M, Krause R et al (2002) Functional organization of the yeast proteome by systematic analysis of protein complexes. Nature 415:141–147
5. Rigaut G, Shevchenko A, Rutz B et al (1999) A generic protein purification method for protein complex characterization and proteome exploration. Nat Biotechnol 17:1030–1032
6. Zhu H, Bilgin M, Bangham R et al (2001) Global analysis of protein activities using proteome chips. Science 293:2101–2105
7. Shoemaker BA, Panchenko AR (2007) Deciphering protein–protein interactions. Part I. Experimental techniques and databases. PLoS Comput Biol 3(3):e42. https://doi.org/10.1371/journal.pcbi.0030042
8. von Mering C, Krause R, Snel B et al (2002) Comparative assessment of large-scale data sets of protein-protein interactions. Nature 417:399–403
9. Yang J, Roy A, Zhang Y (2013) BioLiP: a semi-manually curated database for biologically relevant ligand-protein interactions. Nucleic Acids Res 41:D1096–D1103
10. Schaefer MH, Serrano L, Andrade-Navarro MA (2015) Correcting for the study bias associated with protein–protein interaction measurements reveals differences between protein degree distributions from different cancer types. Front Genet 6:260. https://doi.org/10.3389/fgene.2015.00260
11. Hou Q, Lensink MF, Heringa J et al (2016) CLUB-MARTINI: selecting favourable interactions amongst available candidates, a coarse-grained simulation approach to scoring docking decoys. PLoS One 11:e0155251
12. Hoskins J, Lovell S, Blundell TL (2006) An algorithm for predicting protein-protein interaction sites: abnormally exposed amino acid residues and secondary structure elements. Protein Sci 15:1017–1029
13. Cumberworth A, Lamour G, Babu MM et al (2013) Promiscuity as a functional trait: intrinsically disordered regions as central players of interactomes. Biochem J 454:361–369
14. Altschul SF, Madden TL, Schäffer AA et al (1997) Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. Nucleic Acids Res 25:3389–3402
15. Katoh K, Misawa K, Kuma K-I et al (2002) MAFFT: a novel method for rapid multiple sequence alignment based on fast Fourier transform. Nucleic Acids Res 30:3059–3066
16. Krogh A, Larsson B, von Heijne G et al (2001) Predicting transmembrane protein topology with a hidden Markov model: application to complete genomes. J Mol Biol 305:567–580
17. Krissinel E, Henrick K (2007) Inference of macromolecular assemblies from crystalline state. J Mol Biol 372:774–797

18. Krissinel E (2012) Enhanced fold recognition using efficient short fragment clustering. J Mol Biochem 1:76–85
19. Davis IW, Leaver-Fay A, Chen VB et al (2007) MolProbity: all-atom contacts and structure validation for proteins and nucleic acids. Nucleic Acids Res 35:W375–W383
20. Sali A, Blundell TL (1993) Comparative protein modelling by satisfaction of spatial restraints. J Mol Biol 234:779–815
21. Zhang J, Kurgan L (2019) SCRIBER: accurate and partner type-specific prediction of protein-binding residues from proteins sequences. Bioinformatics 35:i343–i353
22. Murakami Y, Mizuguchi K (2010) Applying the Naïve Bayes classifier with kernel density estimation to the prediction of protein–protein interaction sites. Bioinformatics 26 (15):1841–1848. https://doi.org/10.1093/bioinformatics/btq302
23. Li N, Sun Z, Jiang F (2008) Prediction of protein-protein binding site by using core interface residue and support vector machine. BMC Bioinformatics 9:553
24. Sriwastava BK, Basu S, Maulik U (2015) Protein–protein interaction site prediction in Homo sapiens and E. coli using an interaction-affinity based membership function in fuzzy SVM. J Biosci 40(4):809–818. https://doi.org/10.1007/s12038-015-9564-y
25. Yan C, Dobbs D, Honavar V (2004) A two-stage classifier for identification of protein-protein interface residues. Bioinformatics 20(Suppl 1):i371–i378
26. Hou Q, De Geest PFG, Vranken WF et al (2017) Seeing the trees through the forest: sequence-based homo- and heteromeric protein-protein interaction sites prediction using random forest. Bioinformatics 33:1479–1487
27. Northey TC, Barešić A, Martin ACR (2018) IntPred: a structure-based predictor of protein–protein interaction sites. Bioinformatics 34(2):223–229. https://doi.org/10.1093/bioinformatics/btx585
28. Wang X, Yu B, Ma A et al (2019) Protein–-protein interaction sites prediction by ensemble random forests with synthetic minority oversampling technique. Bioinformatics 35 (14):2395–2402. https://doi.org/10.1093/bioinformatics/bty995
29. Chen H, Zhou H-X (2005) Prediction of interface residues in protein-protein complexes by a consensus neural network method: test against NMR data. Proteins 61:21–35
30. Fariselli P, Pazos F, Valencia A et al (2002) Prediction of protein-protein interaction sites in heterocomplexes with neural networks. Eur J Biochem 269:1356–1361
31. Ofran Y, Rost B (2003) Predicted protein-protein interaction sites from local sequence information. FEBS Lett 544:236–239
32. Porollo A, Meller J (2007) Prediction-based fingerprints of protein-protein interactions. Proteins 66:630–645
33. Zeng M, Zhang F, Wu F-X et al (2020) Protein-protein interaction site prediction through combining local and global features with deep neural networks. Bioinformatics 36:1114–1120
34. Zhang B, Li J, Quan L et al (2019) Sequence-based prediction of protein-protein interaction sites by simplified long short-term memory network. Neurocomputing 357. https://doi.org/10.1016/j.neucom.2019.05.013
35. Li Y, Golding GB, Ilie L (2020) DELPHI: accurate deep ensemble model for protein interaction sites prediction. Bioinformatics btaa750. https://doi.org/10.1101/2020.01.31.929570. https://academic.oup.com/bioinformatics/advance-article-abstract/doi/10.1093/bioinformatics/btaa750/5896983
36. Day B, Cangea C, Jamasb AR, Lió P (2020) Message passing neural processes. https://arxiv.org/abs/2009.13895
37. Gainza P, Sverrisson F, Monti F et al (2020) Deciphering interaction fingerprints from protein molecular surfaces using geometric deep learning. Nat Methods 17:184–192
38. Fout A, Byrd J, Shariat B et al (2017) Protein interface prediction using graph convolutional networks. In: Proceedings of the 31st international conference on neural information processing systems. Curran Associates Inc., Red Hook, NY, pp 6533–6542
39. Sanyal S, Anishchenko I, Dagar A et al (2020) ProteinGCN: protein model quality assessment using graph convolutional networks. Bioinformatics btaa714. https://www.biorxiv.org/content/10.1101/2020.04.06.028266v1
40. Torng W, Altman RB (2019) Graph convolutional neural networks for predicting drug-target interactions. J Chem Inf Model 59:4131–4149
41. Zamora-Resendiz R, Crivelli S (2019) Structural learning of proteins using graph convolutional neural networks. https://doi.org/10.1101/610444
42. Spalević S, Veličković P, Kovačević JNikolić M (2020) Hierarchical protein function

prediction with tail-GNNs. https://arxiv.org/abs/2007.12804
43. de Vries SJ, Bonvin AMJJ (2006) Intramolecular surface contacts contain information about protein-protein interface regions. Bioinformatics 22:2094–2098
44. Martin J (2014) Benchmarking protein-protein interface predictions: why you should care about protein size. Proteins 82:1444–1452
45. Zhang J, Kurgan L (2018) Review and comparative assessment of sequence-based predictors of protein-binding residues. Brief Bioinform 19:821–837
46. Anaconda Software Distribution. Computer software. Vers. 2-2.4.0. Anaconda, Nov 2016. https://www.anaconda.com/
47. Garreta R, Moncecchi G (2013) Learning scikit-learn: machine learning in Python. Packt Publishing Ltd, Birmingham
48. Paszke A, Gross S, Massa F et al (2019) PyTorch: an imperative style, high-performance deep learning library. In: Wallach H, Larochelle H, Beygelzimer A et al (eds) Advances in neural information processing systems 32. Curran Associates Inc, Red Hook, NY, pp 8026–8037
49. Abadi M, Agarwal A, Barham P Brevdo E, Chen Z, Citro C, Corrado GS, Davis A, Dean J, Devin M, Ghemawat S, Goodfellow I, Harp A, Irving G, Isard M, Jia Y, Jozefowicz R, Kaiser L, Kudlur M, Levenberg J, Mane D, Monga R, Moore S, Murray D, Olah C, Schuster M, Shlens J, Steiner B, Sutskever I, Talwar K, Tucker P, Vanhoucke V, Vasudevan V, Viegas F, Vinyals O, Warden P, Wattenberg M, Wicke M, Yu Y, Zheng X (2016) TensorFlow: large-scale machine learning on heterogeneous distributed systems. http://tensorflow.org/
50. Al-Rfou R, Alain G, Almahairi A et al (2016) Theano: a Python framework for fast computation of mathematical expressions. Comput Sci. abs/1605.02688
51. Erickson BJ, Korfiatis P, Akkus Z et al (2017) Toolkits and libraries for deep learning. J Digit Imaging 30:400–405
52. Zhang J, Ma Z, Kurgan L (2019) Comprehensive review and empirical analysis of hallmarks of DNA-, RNA- and protein-binding residues in protein chains. Brief Bioinform 20:1250–1268
53. Hwang H, Pierce B, Mintseris J et al (2008) Protein-protein docking benchmark version 3.0. Proteins 73:705–709
54. Dhole K, Singh G, Pai PP et al (2014) Sequence-based prediction of protein–protein interaction sites with L1-logreg classifier. J Theor Biol 348:47–54. https://doi.org/10.1016/j.jtbi.2014.01.028
55. Jones S, Thornton JM (1997) Prediction of protein-protein interaction sites using patch analysis. J Mol Biol 272:133–143
56. Li ZR, Lin HH, Han LY et al (2006) PROFEAT: a web server for computing structural and physicochemical features of proteins and peptides from amino acid sequence. Nucleic Acids Res 34:W32–W37
57. Zhang P, Tao L, Zeng X et al (2017) PROFEAT update: a protein features web server with added facility to compute network descriptors for studying omics-derived networks. J Mol Biol 429:416–425
58. Cao D-S, Xu Q-S, Liang Y-Z (2013) Propy: a tool to generate various modes of Chou's PseAAC. Bioinformatics 29(7):960–962. https://doi.org/10.1093/bioinformatics/btt072
59. Faraggi E, Zhou Y, Kloczkowski A (2014) Accurate single-sequence prediction of solvent accessible surface area using local and global features. Proteins 82:3170–3176
60. Meiler J, Zeidler A, Schmäschke F et al (2001) Generation and evaluation of dimension-reduced amino acid parameter representations by artificial neural networks. J Mol Model 7:360–369. https://doi.org/10.1007/s008940100038
61. Zimmermann L, Stephens A, Nam S-Z et al (2018) A completely reimplemented MPI bioinformatics toolkit with a new HHpred server at its core. J Mol Biol 430:2237–2243
62. Remmert M, Biegert A, Hauser A et al (2012) HHblits: lightning-fast iterative protein sequence searching by HMM-HMM alignment. Nat Methods 9(2):173–175. https://doi.org/10.1038/nmeth.1818
63. Henikoff S, Henikoff JG (1992) Amino acid substitution matrices from protein blocks. Proc Natl Acad Sci U S A 89:10915–10919
64. Li Y, Ilie L (2017) SPRINT: ultrafast protein-protein interaction prediction of the entire human interactome. BMC Bioinformatics 18:485
65. Dosztányi Z, Mészáros B, Simon I (2009) ANCHOR: web server for predicting protein binding regions in disordered proteins. Bioinformatics 25:2745–2746
66. Kyte J, Doolittle RF (1982) A simple method for displaying the hydropathic character of a protein. J Mol Biol 157:105–132
67. Asgari E, Mofrad MRK (2015) Continuous distributed representation of biological

sequences for deep proteomics and genomics. PLoS One 10:e0141287

68. Alley EC, Khimulya G, Biswas S et al (2019) Unified rational protein engineering with sequence-based deep representation learning. Nat Methods 16:1315–1322
69. Suzek BE, Wang Y, Huang H et al (2015) UniRef clusters: a comprehensive and scalable alternative for improving sequence similarity searches. Bioinformatics 31:926–932
70. Rives A, Meier J, Sercu T , Goyal S, Lin Z, Liu J, Guo D, Ott M, Zitnick CL, Ma J, Fergus R (2021) Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences. Proc Natl Acad Sci U S A 118(15):e2016239118. https://doi.org/10.1073/pnas.2016239118. https://www.pnas.org/content/118/15/e2016239118. https://www.biorxiv.org/content/10.1101/622803v3.abstract
71. Joosten RP, te Beek TAH, Krieger E et al (2011) A series of PDB related databases for everyday needs. Nucleic Acids Res 39: D411–D419
72. Kabsch W, Sander C (1983) Dictionary of protein secondary structure: pattern recognition of hydrogen-bonded and geometrical features. Biopolymers 22:2577–2637
73. Jamasb AR, Lió P, Blundell TL (2020) Graphein—a Python library for geometric deep learning and network analysis on protein structures. https://www.biorxiv.org/content/10.1101/2020.07.15.204701v1.abstract
74. Armstrong DR, Berrisford JM, Conroy MJ et al (2020) PDBe: improved findability of macromolecular structure data in the PDB. Nucleic Acids Res 48:D335–D343
75. Altschul SF, Gish W, Miller W et al (1990) Basic local alignment search tool. J Mol Biol 215:403–410
76. Hubbard TJP, Ailey B, Brenner SE et al (1999) SCOP: a structural classification of proteins database. Nucleic Acids Res 28 (1):257–259. https://doi.org/10.1093/nar/27.1.254
77. Orengo CA, Michie AD, Jones S et al (1997) CATH—a hierarchic classification of protein domain structures. Structure 5 (8):1093–1108. https://doi.org/10.1016/s0969-2126(97)00260-8
78. Kinjo AR, Nishikawa K (2004) Eigenvalue analysis of amino acid substitution matrices reveals a sharp transition of the mode of sequence conservation in proteins. Bioinformatics 20:2504–2508
79. Rost B (1999) Twilight zone of protein sequence alignments. Protein Eng 12 (2):85–94. https://doi.org/10.1093/protein/12.2.85
80. Zhang B, Jaroszewski L, Rychlewski L et al (1997) Similarities and differences between nonhomologous proteins with similar folds: evaluation of threading strategies. Fold Des 2:307–317
81. Torng W, Altman RB (2017) 3D deep convolutional neural networks for amino acid environment similarity analysis. BMC Bioinformatics 18:302
82. Torng W, Altman RB (2019) High precision protein functional site detection using 3D convolutional neural networks. Bioinformatics 35:1503–1512
83. Sato R, Ishida T (2019) Protein model accuracy estimation based on local structure quality assessment using 3D convolutional neural network. PLoS One 14:e0221347
84. Jiménez J, Škalič M, Martínez-Rosell G et al (2018) KDEEP: protein-ligand absolute binding affinity prediction via 3D-convolutional neural networks. J Chem Inf Model 58:287–296
85. Wallach I, Dzamba M, Heifets A (2015) AtomNet: a deep convolutional neural network for bioactivity prediction in structure-based drug discovery. http://arxiv.org/abs/1510.02855
86. Battaglia PW, Hamrick JB, Bapst V, Sanchez-Gonzalez A, Zambaldi V, Malinowski M, Tacchetti A, Raposo D, Santoro A, Faulkner R, Gulcehre C, Song F, Ballard A, Gilmer J, Dahl G, Vaswani A, Allen K, Nash C, Langston V, Dyer C, Heess N, Wierstra D, Kohli P, Botvinick M, Vinyals O, Li Y, Pascanu R (2018) Relational inductive biases, deep learning, and graph networks. Front Artif Intell 4:618372
87. Glorot X, Bengio Y (2010) Understanding the difficulty of training deep feedforward neural networks
88. Chicco D, Jurman G (2020) The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation. BMC Genomics 21:6
89. Ancona M, Ceolini E, Öztireli C, Gross M (2019) Gradient-based attribution methods. In: Samek W, Montavon G, Vedaldi A, Hansen LK, Müller K-R (eds) Explainable AI: interpreting, explaining and visualizing deep learning. Springer, Cham, pp 169–191. ISBN: 978-3-030-28954-6. https://doi.org/10.1007/978-3-030-28954-6_9
90. Vapnik V, Kotz S (2006) Estimation of dependences based on empirical data: empirical

inference science. Information science and statistics. Springer, Berlin. ISBN: 0387308652. https://doi.org/10.1007/0-387-34239-7

91. Abbasi WA, Asif A, Ben-Hur A et al (2018) Learning protein binding affinity using privileged information. BMC Bioinformatics 19:425
92. Chen I, Johansson FD, Sontag D (2018) Why is my classifier discriminatory? https://arxiv.org/abs/1805.12002
93. Amodei D, Olah C, Steinhardt J , Christiano P, Schulman J, Mané D (2016) Concrete problems in AI safety . https://arxiv.org/abs/1606.06565
94. Bernardo JM, Smith AFM (2009) Bayesian theory. Wiley, Hoboken
95. Sverchkov Y, Craven M (2017) A review of active learning approaches to experimental design for uncovering biological networks. PLoS Comput Biol 13:e1005466
96. Deac A, VeliČković P, Sormanni P (2019) Attentive cross-modal paratope prediction. J Comput Biol 26:536–545
97. Huang G, Li Y, Pleiss G, Liu Z, Hopcroft JE, Weinberger KQ (2017) Snapshot ensembles: train 1, get M for free. http://arxiv.org/abs/1704.00109
98. Deng L, Guan J, Dong Q et al (2009) Prediction of protein-protein interaction sites using an ensemble method. BMC Bioinformatics 10:426
99. Pires DEV, Ascher DB, Blundell TL (2014) mCSM: predicting the effects of mutations in proteins using graph-based signatures. Bioinformatics 30:335–342
100. Pires DEV, Blundell TL, Ascher DB (2016) mCSM-lig: quantifying the effects of mutations on protein-small molecule affinity in genetic disease and emergence of drug resistance. Sci Rep 6:29575
101. Blagus R, Lusa L (2013) SMOTE for high-dimensional class-imbalanced data. BMC Bioinformatics 14(1):106. https://doi.org/10.1186/1471-2105-14-106
102. Kingma DP, Ba J (2017) Adam: a method for stochastic optimization. http://arxiv.org/abs/1412.6980
103. Goodfellow I, Bengio Y, Courville A (2016) Deep learning. MIT Press, Cambridge
104. Sun R (2019) Optimization for deep learning: theory and algorithms. http://arxiv.org/abs/1912.08957
105. Bergstra J, Bengio Y (2012) Random search for hyper-parameter optimization. J Mach Learn Res 13:281–305
106. Snoek J, Larochelle H, Adams RP (2012) Practical Bayesian optimization of machine learning algorithms. In: Pereira F, Burges CJC, Bottou L et al (eds) Advances in neural information processing systems 25. Curran Associates, Inc, Red Hook, NY, pp 2951–2959