



# MONASH University

## **Protein sequential, structural and functional analysis using deep learning**

by

**Ying Xu,**  
MSciEng, MIT

A thesis submitted for the degree of Doctor of Philosophy at  
Monash University in 2018  
Faculty of Information Technology

## **Copyright notice**

### **Notice 1**

Under the Copyright Act 1968, this thesis must be used only under the normal conditions of scholarly fair dealing. In particular no results or conclusions should be extracted from it, nor should it be copied or closely paraphrased in whole or in part without the written consent of the author. Proper written acknowledgement should be made for any assistance obtained from this thesis.

### **Notice 2**

I certify that I have made all reasonable efforts to secure copyright permissions for third-party content included in this thesis and have not knowingly added copyright content to my work without the owner's permission.

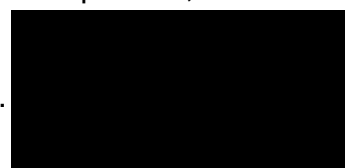
## Declaration

I declared that this thesis contains no material which has been accepted for the award of any other degree or diploma at any university or equivalent institution and that, to the best of my knowledge and belief, this thesis contains no material previously published or written by another person, except where due reference is made in the text of the thesis.

Print Name: YING XU

Date: April 20<sup>th</sup>, 2018

Signature: ...



## Acknowledgements

The research in this thesis could not have been performed if not for the assistance, patience and help of many individuals. Here, I would like to extend my gratitude to my supervisors: Dr. Campbell Wilson, Dr. Jiangning Song, Dr. Jue (Grace) Xie, and Prof. James Whisstock. They have been devoted extensive patience and professional supervisions during the last three years.

My great thankfulness goes to Campbell for his support in every aspect of this project. Thanks to Campbell, I enjoyed the whole journey with the most possible flexibility in exploring every possible aspect of the research topic and its extended topics. He also provided the most generous help in academic supervision, professional suggestion and mental health consultancy.

My great thankfulness also goes to Jiangning for his tremendous time and effort spent in my academic training and supervision. Besides his bioinformatics expertise, he also contributed by training me to be a rigorous researcher and teaching me how to strategically plan our research works and publications.

I gratefully thank James for his insights and guidance from a higher level of view, which makes the research works in this thesis more profound from the perspective of bioinformatics and biology. I sincerely respect his expertise in structural biology.

I would like to thank Grace for her academic supervision and professional suggestions during the whole project. She was the first one who strongly encouraged me to pursue my PhD study, after which she has long been my supervisor both professionally and personally.

I would like to thank Julie Holden, who provided professional suggestions in thesis writing and proofread my thesis with the most patience. Also, I would like to thank Dr. Jey Han Lau from the IBM Research, who provided professional supervision about deep learning and generously shared his experience in the application of deep learning to natural language processing. In addition, I would like to thank Dr. Tali Boneh, Prof. Ann Nicolson, and Dr. Kevin Korb for their supervision in performing quantitative evaluation of prediction models. My sincere



thankfulness also goes to Prof. André Leier and Prof. Tatiana T. Marquez-Lago for their proofread and comments for our publication drafts. Specifically, I would like to thank Jerico Revote for his help in maintaining computing resources that are used in this thesis. Without his help, I would not have been able to deliver our online web services.

I would also extend my thankfulness to my colleagues and fellow PhD students, Yuxin Zhang, Tenny Yuan, Yuri Song, Dora He, Ariesta Lestari, Prajwol Sangat, and Tian Gohs, and to the academic and administrative staff, Prof. Frada Burstein, Prof. Bala Srinivasan, Dr. Maria Indrawan, Dr. Pari Delir Haghighi, Helen Cridland and Allison Mitchell, who made this PhD journey more enjoyable and helpful.

I would like to express my deepest gratefulness to my family – Grandma, Dad, Mum, aunties and uncles - who are on the other side the world. They have always been so supportive no matter what decision I have made. It was not easy for them, but they tried their best to respect me, to understand me, and to support me in every way they can. I understand that, without their support and love, I would not have been this far in my life.

Special thanks to my partner, Chris James, who has always been on my side in the last three years, taking care of me and making me laugh. Thanks to him and his family, I felt less homesick.

I also would like to acknowledge the financial support from the Monash Graduate Scholarship (MGS) and the Monash International Postgraduate Research Scholarship (MIPRS).

## Abstract

Biological data has been generated at an unprecedented rate during the last 20 years due to the rapid development of different digital automatic sensors. While the increase in the amount of available data is exciting, it has become more challenging for biologists to analyse, annotate and interpret the generated biological data. Accordingly, computational methods that combine techniques from machine learning, data mining, mathematics and statistics are increasingly developed to facilitate the annotation of the biological data and provide biological interpretations and insights.

This thesis focuses on the analysis of proteins, the most important functional product of gene expression. The large amount of protein sequences and the limited quantities of characterised protein structures and functions leads to a compelling research topic, which is concerned with the computationally prediction of protein structures and functions directly from protein sequences. Existing computational methods for protein structural/functional prediction are mainly machine learning-based, implemented using shallow architectures that are inferior in expressive power when compared with more recent deep learning architectures. Despite the state-of-the-art performance achieved by deep learning in protein analysis in the last three years, existing research is conducted in an *ad hoc* manner, suffering from a lack of systematic design of deep learning frameworks for groups of prediction tasks. Critically, existing deep learning models for protein prediction analysis are designed without incorporating the biological background knowledge, an important source of heuristics.

In this thesis, we conduct comprehensive benchmarking experiments and assessment to address research questions. Firstly, how to effectively represent protein sequences for different granularities of prediction tasks and, specifically, how do purely data-driven representations perform compared with biological heuristic-based representations? Secondly, how to design a deep learning framework that can be applied to predict different protein structural and functional properties? Thirdly, how to leverage limited quantities of labelled biological data for rigorous model training using biological background knowledge?

To answer the above three research questions, four deep learning frameworks are proposed. The generation and application framework of *the distributed representation of protein sequences* are proposed for addressing sequence-level, residue-level and biological interaction-level prediction tasks. A deep learning framework, *DeepS2P*, is designed as a framework that can be applied to any protein sequence-based residue-level prediction tasks. Multitask frameworks, *Multi-task and Cross-stitch*, are proposed based on the observed qualitative correlation between protein structural properties, and in turn are used to predict two or more structural properties simultaneously with quantitative correlation discovery. Finally, the deep transfer learning framework, *DeepTransfer*, is specifically designed for predicting functionally important sites of enzymes organised in a hierarchy. The rationale of this framework is to incorporate the knowledge of hierarchical classification systems in biology to balance out the requirement for large quantities of training data.

The proposed deep learning frameworks are evaluated in their applications to five specific prediction tasks, including protein sequence-level prediction of protein families; residue-level prediction of intrinsically disordered proteins and regions, protein secondary structure populations, and phosphorylation sites; and biological interaction-level prediction of drug-target interactions. For related prediction tasks, bioinformatics tools are developed for public access and quantitative analyses are conducted to critically investigate the factors that may affect the prediction performance. To summarise, a set of heuristics for designing deep learning frameworks in protein analysis is proposed based on the proposed frameworks, the application scenarios and the evaluation results.

## **Thesis publications**

Xu, Y., Song, J., Wilson, C.C., Whisstock, J.C. PhosContext2vec: a distributed representation of residue-level sequence contexts and its application to general and kinase-specific phosphorylation site prediction. Scientific Reports, submitted and revised.

# Table of Contents

<b>1</b>	<b>Introduction .....</b>	<b>1</b>
1.1	Preamble .....	1
1.2	An introduction to the biological background.....	3
1.2.1	Gene expression and protein expression .....	3
1.2.2	Protein sequences, structures and functions.....	6
1.2.3	Protein structural and functional properties .....	8
1.2.4	Databases for protein analysis .....	14
1.2.5	Bioinformatics tools for protein analysis .....	15
1.3	An introduction to the background of deep learning .....	16
1.3.1	The history and development of deep learning .....	16
1.3.2	An introduction to different deep learning models .....	17
1.3.3	Advantages and disadvantages of deep learning.....	19
1.4	Existing challenges, research questions and objectives .....	20
1.4.1	Research question #1.....	21
1.4.2	Research question #2.....	22
1.4.3	Research question #3.....	22
1.5	Contributions and scope.....	23
1.6	An overview of the thesis structure.....	24
<b>2</b>	<b>Literature Review .....</b>	<b>28</b>
2.1	Deep learning models, strategies and techniques.....	28
2.1.1	Basics of machine learning and neural networks .....	28
2.1.2	Deep learning models.....	32
2.1.3	Distributed representation of words, sentences and documents.....	36
2.1.4	Transfer learning, multitask learning and deep learning.....	40
2.1.5	Open source platforms for constructing deep learning models .....	42
2.2	Protein analysis based on deep learning techniques .....	43
2.2.1	Existing representations of protein sequences.....	43
2.2.2	Limitations of the application of existing protein representations .....	47
2.2.3	Existing applications of deep learning to protein analysis .....	48
2.2.4	Limitations of existing applications of deep learning to protein analysis ....	51
2.3	Existing methods for protein prediction tasks .....	51
2.3.1	Protein family prediction .....	52
2.3.2	Intrinsically disordered protein prediction .....	54
2.3.3	Phosphorylation site prediction.....	58
2.3.4	Drug-target interaction prediction .....	61
2.4	Chapter summary .....	71

<b>3</b>	<b>Scientific method .....</b>	<b>73</b>
3.1	Scientific method in general .....	73
3.2	Data Collection .....	74
3.2.1	The construction of training dataset and testing dataset.....	75
3.2.2	The collection of labelled data and the construction of input features.....	76
3.3	Model training and model application .....	76
3.3.1	The cross-validation test.....	78
3.3.2	The independent test.....	78
3.3.3	Vector space analysis.....	79
3.3.4	Correlation analysis .....	79
3.3.5	Case study.....	80
3.4	Performance Evaluation .....	80
3.5	Result interpretation .....	82
3.6	Chapter summary .....	82
<b>4</b>	<b>Deep learning frameworks for protein analysis .....</b>	<b>84</b>
4.1	The distributed representation of protein sequences .....	84
4.1.1	Notation system for protein sequences .....	85
4.1.2	Prot2vec: the distributed representation of protein sequences .....	86
4.1.3	The training of the prot2vec generative models .....	95
4.1.4	The application framework of prot2vecs.....	99
4.1.5	Conclusion.....	104
4.2	Deep learning framework for protein sequence-based predictions .....	105
4.2.1	Protein structural and functional properties .....	106
4.2.2	Protein sequence-based residue-level predictions.....	107
4.2.3	Feature selection, construction and extraction.....	108
4.2.4	DeepS2P: a deep learning framework for protein sequence-based predictions .....	113
4.2.5	Hyper-parameters for DeepS2P .....	116
4.2.6	Conclusion.....	119
4.3	Multitask deep learning for protein structural prediction .....	119
4.3.1	Qualitative correlation between protein structural properties .....	120
4.3.2	Multitask deep learning with hard parameter sharing.....	121
4.3.3	Cross-stitch multitask learning for quantitative correlation analysis .....	122
4.3.4	Alternate training .....	124
4.3.5	Conclusion.....	125
4.4	Deep transfer learning for functionally important site prediction.....	125
4.4.1	Post-translational modification site prediction .....	125
4.4.2	Hierarchical data structures of proteins and enzymes.....	126
4.4.3	Hidden layers as feature extractors.....	127

4.4.4	Deep transfer learning for hierarchical enzymes .....	128
4.4.5	Overfitting and underfitting .....	129
4.4.6	Conclusion .....	129
4.5	Chapter summary .....	130
<b>5</b>	<b>Protein family prediction .....</b>	<b>132</b>
5.1	Introduction .....	132
5.2	An overview of protein family prediction .....	132
5.3	Dataset construction .....	134
5.3.1	The UniProt-Pfam dataset .....	135
5.3.2	The UniProt-InterPro dataset .....	135
5.4	Vector space analysis for human protein families .....	136
5.4.1	Experimental design .....	136
5.4.2	Results analysis .....	137
5.5	Comparison with existing protein representations .....	142
5.5.1	Comparison with existing distributed representations .....	143
5.5.2	Comparison with physiochemical properties-based features .....	147
5.5.3	Comparison with homology profile-based features .....	151
5.6	Investigation of potential factors that affect the prediction performance of PfamProt2vec .....	154
5.6.1	The effect of the organism of training examples .....	154
5.6.2	The effect of functional categories of protein families .....	157
5.6.3	The effect of eight manually selected factors .....	161
5.7	Case study: Tyrosine kinases and Protein kinase domain .....	171
5.8	The implementation of the online webserver .....	174
5.9	Chapter summary .....	175
<b>6</b>	<b>Protein IDP/IDR and SSP prediction .....</b>	<b>177</b>
6.1	Introduction .....	177
6.2	Protein intrinsic disorder prediction based on the DeepS2P framework ...	177
6.2.1	An overview of IDP/IDR prediction .....	178
6.2.2	Experimental design .....	179
6.2.3	Determination of hyper-parameters .....	180
6.2.4	Comparison between deep architectures and shallow architectures .....	182
6.2.5	Conclusion .....	187
6.3	Protein SSP prediction based on the DeepS2P framework .....	187
6.3.1	An overview of SSP prediction .....	187
6.3.2	The difference between protein SSP prediction and SS prediction .....	188
6.3.3	Experimental design .....	189
6.3.4	Determination of hyper-parameters .....	191

6.3.5	Comparing linear models, non-linear shallow models and deep models .	193
6.3.6	Validation of DeepS2P-P on disordered proteins and regions .....	194
6.3.7	Validation of DeepS2P-P on benchmark datasets .....	195
6.3.8	Conclusion.....	196
6.4	Simultaneous prediction of IDP/IDR and SSP using multitask frameworks	197
6.4.1	Experimental design .....	198
6.4.2	Comparison between single-task models and multitask models .....	198
6.4.3	Independent test on CASP 9 and CASP 10 targets .....	201
6.4.4	Quantitative correlation between IDP/IDR and SSP predictions .....	203
6.4.5	Case study on T520, p53 and SOSSB1 .....	204
6.4.6	Conclusion.....	209
6.5	Chapter summary .....	209
<b>7</b>	<b>Phosphorylation site prediction .....</b>	<b>211</b>
7.1	Introduction.....	211
7.2	Protein kinase and its phosphorylation sites .....	211
7.3	An overview of phosphorylation site prediction .....	212
7.4	Phosphorylation site prediction based on context2vecs.....	214
7.4.1	Selected residue-level features for phosphorylation site prediction .....	215
7.4.2	Generation of context2vecs for phosphorylation site prediction .....	216
7.4.3	Experimental design .....	218
7.4.4	The effect of incorporating the distributed contextual feature vectors .....	221
7.4.5	The comparison between two implementations of prot2vecs.....	227
7.4.6	Determination of hyper-parameters.....	230
7.4.7	Independent test for general phosphorylation site prediction .....	233
7.4.8	Independent test for kinase-specific phosphorylation site prediction .....	236
7.4.9	Comparison of different method in terms of time efficiency.....	244
7.4.10	Implementation of the PhosContext2vec web server .....	245
7.4.11	Conclusion.....	247
7.5	Phosphorylation site prediction based on the DeepS2P framework.....	248
7.5.1	Introduction.....	248
7.5.2	Experimental design .....	248
7.5.3	Determination of hyper-parameters.....	249
7.5.4	Independent test in kinase-specific phosphorylation site prediction.....	252
7.5.5	Conclusion.....	256
7.6	Phosphorylation site prediction using DeepTransfer .....	256
7.6.1	Introduction.....	256
7.6.2	The application of DeepTransfer to phosphorylation site prediction.....	257
7.6.3	Predictive models for kinase nodes in hierarchy .....	259



7.6.4	Experimental design .....	260
7.6.5	Hyper-parameter tuning.....	261
7.6.6	Results for protein kinases in different groups .....	262
7.6.7	Results for groups, families and subfamilies .....	264
7.6.8	Performance improvement for kinases with insufficient annotations.....	266
7.6.9	Vector space analysis for features generated by hidden layers .....	267
7.6.10	Comparison with state-of-the-art methods .....	270
7.6.11	Validation of proposed methods for small datasets.....	274
7.6.12	Conclusion.....	275
7.7	Chapter Summary .....	276
<b>8</b>	<b>Drug-target interaction prediction .....</b>	<b>279</b>
8.1	Introduction.....	279
8.2	An overview of drug-target interaction prediction .....	279
8.3	Experimental design .....	280
8.3.1	Data sources and datasets.....	280
8.3.2	Cross-validation under four experimental settings .....	282
8.3.3	Performance evaluation.....	284
8.4	DTI prediction based on existing machine learning methods .....	284
8.4.1	Comparison among machine learning based methods .....	284
8.4.2	Comparison among existing drug/protein kernels .....	288
8.4.3	New interaction prediction with existing methods and kernels .....	290
8.5	DTI prediction with prot2vec-based protein kernels .....	291
8.5.1	Comparison among different kernels generated from prot2vec .....	292
8.5.2	Comparison among three implementations of prot2vecs .....	295
8.5.3	Comparison with the state-of-the-art protein kernels.....	297
8.6	Chapter summary .....	299
<b>9</b>	<b>Conclusions and future work.....</b>	<b>301</b>
9.1	Heuristics for designing deep learning frameworks.....	301
9.1.1	Heuristic 1: Purely data-driven representation should be used in combination with biological heuristic-based representations.....	302
9.1.2	Heuristic 2: Biological background knowledge should be incorporated in the designing of deep learning frameworks for protein and biological predictive analysis.....	303
9.1.3	Heuristic 3: Deep learning frameworks for protein analysis should be designed at a conceptual level .....	305
9.2	Future works.....	306
<b>Appendix A</b>		<b>329</b>
<b>GLOSSARY</b>		<b>339</b>



# List of Figures

<b>Figure 1.1</b> Gene expression with transcription, post-transcriptional modification, translation, post-translational modification, and protein folding.....	5
<b>Figure 1.2</b> The increasing numbers of manually reviewed protein sequences in the Swiss-Prot/UniProt database and solved protein structures in the RCSB PDB database. ....	8
<b>Figure 1.3</b> The thesis structures with respect to chapters and sections. ....	26
<b>Figure 2.1</b> The architecture of a basic neuron in artificial neural networks. ....	31
<b>Figure 2.2</b> Receptive field, convolutional field and ZERO padding in convolutional neural networks.....	33
<b>Figure 2.3</b> Consecutive units in recurrent neural networks. ....	35
<b>Figure 2.4</b> The mappings between natural language processing and biological sequence analysis. ....	36
<b>Figure 2.5</b> The distributed representation of words and phrases [170]. ....	38
<b>Figure 2.6</b> The distributed representation of sentences and documents [169]. ....	39
<b>Figure 2.7</b> The DeepCNF-SS model for protein secondary structure prediction [156]. ....	49
<b>Figure 2.8</b> Bayesian matrix factorization for drug-target interaction prediction [233]. ....	66
<b>Figure 2.9</b> An example of applying semantic-based edge partitioning to drug-target interaction prediction. ....	68
<b>Figure 2.10</b> RBM template for predicting interactions of individual protein $t$ [253]. ....	69
<b>Figure 3.1</b> An overview of the scientific method that is used in this thesis. ....	74
<b>Figure 3.2</b> The roles of training and testing datasets in the application of machine learning predictive models. ....	75
<b>Figure 4.1</b> Unigram and 3-gram splitting with overlapping and non-overlapping strategies for the amino acid sequence of protein HUNIN_HUMAN. ....	89
<b>Figure 4.2</b> Difference implementations of prot2vec in the training and inference step. ....	93
<b>Figure 4.3</b> The application framework of prot2vecs in three different scenarios.....	105
<b>Figure 4.4</b> Protein sequence, protein structural properties and protein functional properties..	107
<b>Figure 4.5</b> The difference between the architecture of DeepS2P and multitask framework with hard-parameter sharing. ....	121
<b>Figure 4.6</b> The framework for a cross-stitch multitask deep convolutional neural network.....	123
<b>Figure 4.7</b> Deep transfer learning for kinase-specific phosphorylation site prediction. ....	128

<b>Figure 5.1</b> The searched protein domain for protein EGFR_HUMAN and NGAL_HUMAN returned by CDD.....	133
<b>Figure 5.2</b> Feature vector analysis for the top six PF pairs with minimum cosine distances. ...	139
<b>Figure 5.3</b> Feature vector space analysis of PF pairs with maximum cosine distances. ....	140
<b>Figure 5.4</b> Prediction performance of all protein families between prot2vec <sup>inference</sup> and prot2vec <sup>add</sup> . ....	144
<b>Figure 5.5</b> Prediction performance of Prot2vecDM for five common protein families shared across human, mouse, A. thaliana and all organisms. ....	156
<b>Figure 5.6</b> The correlation between the eight potential factors and the prediction accuracy of PfamProt2vec (red) and HMMER 3.1 (blue). ....	165
<b>Figure 5.7</b> Phylogenetic trees of the protein families with the worse prediction accuracies (left) and the protein families with the best prediction accuracies (right). ....	169
<b>Figure 5.8</b> Independent test in protein family Tyrosine kinase (PF07714), trained with human proteins and all proteins respectively. ....	172
<b>Figure 6.1</b> Protein secondary structure populations of structured and unstructured proteins and regions. ....	195
<b>Figure 6.2</b> Prediction performance of deep learning models for IDP/IDR prediction using 10-fold cross-validation. ....	200
<b>Figure 6.3</b> Cross-stitch units indicating a linear correlation between IDP/IDR prediction (task A) and SSP prediction (task B). ....	204
<b>Figure 6.4</b> Prediction results for target T0520 in CASP9 by DeepS2P-D, Multi-task-D, PSIPRED, DISOPRED, PrDOS2 and s2D. ....	205
<b>Figure 6.5</b> Prediction results for p53 and SOSSB1 by DeepS2P-D and Multi-task-D. ....	207
<b>Figure 7.1</b> Extraction of residue-level feature groups and the generation of the distributed contextual feature vector for phosphorylation site prediction. ....	217
<b>Figure 7.2</b> Performance comparison between models trained based on context2vec <sup>add</sup> and context2vec <sup>inference</sup> feature vectors for (a) <b>general</b> and (b) <b>kinase-specific</b> phosphorylation site prediction, evaluated in terms of AUC score. ....	228
<b>Figure 7.3</b> Hyper-parameter tuning for SVM models trained with distributed contextual feature vectors for (a) <b>general</b> and (b) <b>kinase-specific</b> phosphorylation site prediction. ....	232
<b>Figure 7.4</b> ROC curves of PhosContext2Vec and four existing methods on the independent test for general phosphorylation site prediction. ....	234
<b>Figure 7.5</b> ROC curves of PhosContext2vec and seven existing methods for kinase-specific phosphorylation site prediction on the independent test. ....	238

<b>Figure 7.6</b> Elapsed time comparison between different phosphorylation site predictors for predicting 200 protein sequences. ....	244
<b>Figure 7.7</b> The architecture of the PhosContext2vec web server. ....	246
<b>Figure 7.8</b> Hyper-parameter tuning for DeepS2P-Ph models in predicting phosphosites of AGC/PKA, AGC/PKC, CMGC/CDK, Other/CK2, and TK/Src. ....	250
<b>Figure 7.9</b> Hyper-parameter tuning for DeepS2P-Ph models in predicting phosphosites of CAMK, AGC/PKC, CMGC/CDK/CDK5s, and Other/PLK/PLK-/PLK1. ....	251
<b>Figure 7.10</b> Prediction performance comparison of DeepS2P-Ph in cross-validation (training) and independent (testing) tests. ....	255
<b>Figure 7.11</b> Derived models for kinase nodes in hierarchies of the HKCT. ....	259
<b>Figure 7.12</b> Prediction performance of direct models m11 ~ m15 with 1~5 hidden layers for PKs, evaluated using AUC scores. ....	262
<b>Figure 7.13</b> Prediction performance of models for groups, families, subfamilies and PKs in two paths. ....	264
<b>Figure 7.14</b> The correlation between the <i>PIR</i> and the number of annotated phosphosites. ....	267
<b>Figure 7.15</b> The t-SNE plot of vector representations generated from hidden layers. ....	269
<b>Figure 7.16</b> Prediction performance comparison between PhosTransfer and several existing kinase-specific phosphorylation site prediction methods. ....	273
<b>Figure 8.1</b> Relationships between different drug-target interaction data sources. ....	281
<b>Figure 8.2</b> Drug-target interaction prediction under four experimental settings. ....	283
<b>Figure 8.3</b> Comparison among different DTI drug/protein kernel pairs. ....	289
<b>Figure 8.4</b> Performance comparison among different kernels that were generated from prot2vecs. ....	293
<b>Figure 8.5</b> Performance comparison among the three implementations of prot2vecs as the kernel feature vectors for DTI prediction. ....	296
<b>Figure 8.6</b> Performance comparison among state-of-the-art protein kernels for drug-target interaction prediction. ....	298

## List of Tables

<b>Table 1.1</b> A summary of the protein structural and functional properties. ....	13
<b>Table 2.1</b> A summary of machine learning methods for IDP/IDR prediction. ....	57
<b>Table 2.2</b> A summary of machine learning methods for phosphorylation site prediction. ....	60
<b>Table 2.3</b> Four categories of features for drug-target interaction prediction. ....	64
<b>Table 2.4</b> A summary of machine learning based methods for DTI prediction. ....	70
<b>Table 4.1</b> The 1-letter and 2-letter denotations for amino acids (AA). ....	85
<b>Table 4.2</b> Different distributed representation of protein sequences. ....	94
<b>Table 4.3</b> The hyper-parameter tuning for the generative models of prot2vec <sup>inference</sup> and prot2vec <sup>non-overlap</sup> . ....	97
<b>Table 4.4</b> A summary of the input features for the prediction of protein structural and functional properties. ....	110
<b>Table 4.5</b> A summary of the hyper-parameters for the DeepS2P framework. ....	118
<b>Table 4.6</b> A summary of deep learning frameworks and their application scenarios. ....	131
<b>Table 5.1</b> Top rankings of the normalized average cosine distance between protein family pairs. ....	138
<b>Table 5.2</b> The average predictive performance of the prot2vec <sup>inference</sup> -based PfamProt2vec and the prot2vec <sup>add</sup> -based PfamProt2vec in different subsets of protein families. ....	146
<b>Table 5.3</b> Statistics of the constructed training, test and independent test datasets for 10 representative protein functional families. ....	149
<b>Table 5.4</b> Performance comparison between PfamProt2vec and SVM-Prot 2016 for 10 representative families. ....	150
<b>Table 5.5</b> Performance comparison between PfamProt2vec and HMMER 3.1 for protein family prediction in corresponding families. ....	153
<b>Table 5.6</b> Prediction performance of PfamProt2vec for predicting 25 human protein families according to eight functional categories. ....	159
<b>Table 5.7</b> Differences between HMMER 3.1 and PfamProt2vec in terms of methodology. ....	161
<b>Table 5.8</b> Protein families with the best and the worse prediction accuracies achieved by PfamProt2vec. ....	168
<b>Table 5.9</b> Statistics of the constructed cross-validation and independent test datasets for protein family <i>Tyrosine kinase</i> (PF07714) and <i>Protein kinase family</i> (PF00069). ....	172

<b>Table 6.1</b> Hyper-parameter tuning for DeepS2P-D in IDP/IDR prediction.....	181
<b>Table 6.2</b> Performance comparison for IDP/IDR prediction applied to CASP9 proteins. ....	184
<b>Table 6.3</b> Performance comparison for IDP/IDR prediction applied to CASP10 proteins. ....	185
<b>Table 6.4</b> The selection of the hyper-parameters for the single-task deep learning framework. .....	192
<b>Table 6.5</b> Prediction performance comparison between s2D, LR, and DeepS2P-P for protein SSP prediction on the $\delta$ 2D validation set.....	194
<b>Table 6.6</b> Performance comparison between DeepS2P-P and the s2D method on the independent benchmark BMR_2018. ....	196
<b>Table 6.7</b> Performance comparison for IDP/IDR prediction (CASP9). ....	202
<b>Table 7.1</b> Statistical summary of the curated datasets for the five kinases with the most number of annotations. ....	220
<b>Table 7.2</b> Performance comparison between models trained with and without distributed contextual feature vectors for general and kinase-specific phosphorylation site prediction. ....	223
<b>Table 7.3</b> Performance comparison between different predictors of general phosphorylation sites, evaluated in terms of Sensitivity (SE), Specificity (SP) and Matthews correlation coefficient (MCC). ....	241
<b>Table 7.4</b> Performance comparison between different predictors of kinase-specific phosphorylation sites, evaluated in terms of Sensitivity (SE), Specificity (SP) and the Matthews correlation coefficient (MCC). ....	242
<b>Table 7.5</b> Statistics of the training and testing datasets for the four kinase nodes representing different levels of abstraction. ....	249
<b>Table 7.6</b> Prediction performance comparison between DeepS2P-Ph and existing phosphorylation site prediction methods in terms of AUC scores.....	253
<b>Table 7.7</b> Prediction performance comparison between PhosTransfer and GPS 3.0.....	271
<b>Table 7.8</b> Prediction performance of PhosContext2vec, DeepS2P-Ph and PhosTransfer on ST/Other/PLK-/PLK1 in terms of AUC score. ....	274
<b>Table 8.1</b> Statistical summary of proteins, drugs and DTIs in the KEGG, Yamanishi, and Perlman datasets .....	282
<b>Table 8.2</b> Performance comparison among different machine learning methods for drug-target interaction prediction on the Yamanish dataset .....	286
<b>Table 8.3</b> Performance of different machine learning methods for DTI prediction under four experimental settings, evaluated using TP/P and precision (PR). ....	291





# 1 Introduction

## 1.1 Preamble

In recent years, a large amount of biological data has been produced at an unprecedented rate due to the development of new biotechnology [1]. The massive amount of biological data and the high speed of the increase in the volume of such data make it a very challenging task for biologists to draw interpretations and insights efficiently. Therefore, various computational methods have been developed to interpret biological data in large scales, which inspired the interdisciplinary field called *bioinformatics* or *computational biology* [2]. Computational interpretation of biological data is categorised into two types, *i.e.* the *statistical analysis* which draws conclusions that explain biological phenomena, and the *predictive analysis* which makes novel predictions to be experimentally validated. In general, predictive analyses rely on the application of various machine learning-based methods. However, machine learning models with limited expressive power [3] are not necessarily suitable for modelling prediction tasks in biology, due to the complex nature of the biological events such as gene expression [4]. This existing challenge inspires the application of deep learning, a branch of machine learning that has achieved state-of-the-art performance in various application areas [5, 6], to the analysis of the increasing amount of biological data.

The application of deep learning in biological data analysis has achieved state-of-the-art performance in various prediction tasks including protein structure prediction [7-9], alternative splicing [10] and single-cell DNA methylation states [11]. These applications are mostly conducted in an *ad hoc* manner, with a lack of systematic design that incorporates heuristics from the biological background and critical evaluations of the performance of deep learning in predicting biological data at different granularities. In this thesis, we seek to answer some of the more abstract questions, including 1) what

makes it especially challenging to apply deep learning in the area of biology, compared to some of the most popular application areas such as image processing [5], natural language processing [12] and speech recognition [13]; 2) what transferrable knowledge can be drawn from these application areas to assist in designing deep learning models for biological data analysis; 3) what are the relationships between different prediction tasks in biological data analysis and 4) is it possible to design meta deep learning frameworks for groups of prediction tasks. To answer these questions, we propose deep learning frameworks, conduct extensive experiments for various prediction tasks, and produce useful insights for assisting the designing of future deep learning models for biological data analysis.

From the perspective of biology, we focus our study on protein analysis. Protein represents one of the most important products of gene expression and plays crucial roles in various biological events such as enzyme catalysis [14, 15], molecule transportation [16] and DNA binding [17]. Protein functions rely on protein tertiary structures, which are further determined by the protein primary structure – *amino acid sequences (protein sequences)* [18]. Due to the large amount of identified protein amino acid sequences and the limited number of determined protein structures and functions, it is crucial to develop computational methods that can predict novel protein structures and functions directly from protein sequences.

From the perspective of deep learning, we aim to explore the different designs of deep learning frameworks by incorporating the heuristics and domain knowledge from biology. Deep learning represents one of the most popular machine learning methods in the last five years. Its successful applications in image processing, natural language process and speech recognition inspired a large amount of research works in various application areas [6, 11, 19-21]. However, it is crucial to notice the difference between biological data and data from other areas. On one hand, compared to the large amount of image data [22] and corpus data [23] that is publicly available on the Internet, the acquisition of biological data is relatively expensive and labour-intensive. On the other

hand, despite the lack of annotated data, biology represents one of the most developed field in science which, over the time, has a large amount of domain knowledge accumulated. Therefore, a focus of this thesis is to design deep learning frameworks by incorporating heuristics and domain knowledge from biology.

The rest of this chapter is organised as follows. **Section 1.2** and **Section 1.3** provide the biological background and the computational background of the thesis, respectively. **Section 1.4** outlines the challenges, research questions and objectives. **Section 1.5** depicts the main contributions and the research scope. Finally, **Section 1.6** provides an overview over the structure of the whole thesis.

## 1.2 An introduction to the biological background

Bioinformatics is composed of two aspects, bio- and -informatics, which indicates that any research topic in bioinformatics must comply with both the biological background and the informatics background. For a better understanding of the application of deep learning in protein analysis, in this section, we provide an overall introduction of the biological background of protein sequences, structures and functions.

### 1.2.1 Gene expression and protein expression

Life is the product of evolution where changes in genetic materials such as DNAs and RNAs cause the changes in their biological products such as proteins [24]. However, it is a complicated process to synthesise the final biological products, given a specific segment of a gene. This process of synthesising genes into their functional gene products is called *Gene Expression*<sup>1</sup>.

For prokaryotes, gene expression has two main steps, *i.e.* transcription from DNA to *message RNA (mRNA)* and translation from mRNA to peptides or proteins [25]. While

---

<sup>1</sup> Gene expression: [https://en.wikipedia.org/wiki/Gene\\_expression](https://en.wikipedia.org/wiki/Gene_expression)

for eukaryotes, the process of gene expression is composed of at least six steps, as is described below.

**Transcription.** *Transcription* is the first process in the cascade of events that lead from the genetic code contained in DNA to synthesis of a specific protein [26]. it involves synthesising mRNAs from DNAs [25]. DNAs are composed of two reversely complementary nucleotides strands that start from the 3' end and the 5' end respectively. mRNA sequences are copied from DNA strands by adding one RNA nucleotides at a time, resulting in sequences that are complementary to the 3'→ 5' DNA strands and identical to the 5'→ 3' DNA strands [27].

**Post-transcriptional modification.** *Post-transcriptional modification* refer to changes that occur to a newly transcribed primary RNA transcript after transcription has occurred and prior to its translation into a protein product [28]. It includes RNA splicing [29], 5' capping, 3' cleavage and polyadenylation [30].

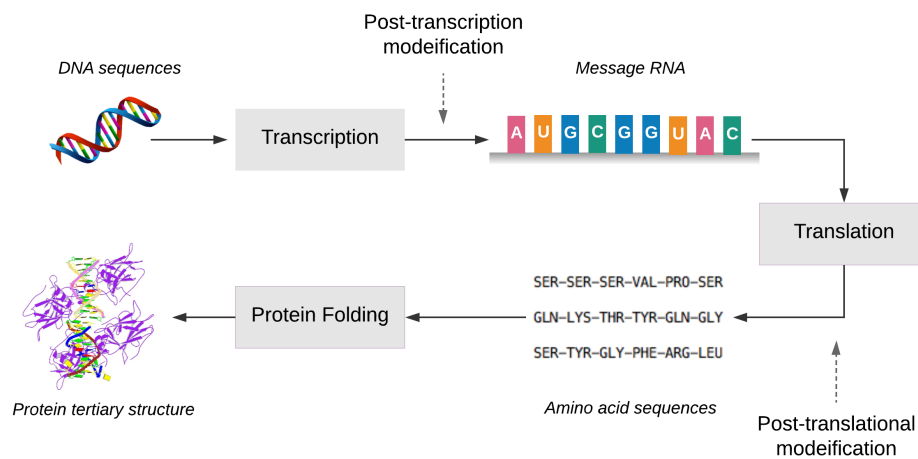
**Translation.** *Translation* is the process by which a protein is synthesized from the information contained in a molecule of messenger RNA (mRNA) [31]. In mRNAs, the coding region, namely the codon, are composed of triplets, which corresponds to binding sites in *transfer RNAs (tRNA)* that are called the anticodon [32]. Protein, polypeptide, and peptide sequences are synthesised by amino acids carried by tRNAs according to the order of the codons in mRNA coding regions.

**Post-translational modification.** *Post-translational modification (PTM)* refers to the covalent and enzymatic modifications after translation [33]. It is a crucial process for maturing the synthesised peptides and is closely related to multiple protein functions. It usually involves the modification of existing functional groups or addition of other functional groups such as phosphate (in phosphorylation) [34], carbohydrate (in glycosylation) and lipid (in lipidation) molecules. *Phosphorylation* is the most common post-translational modification [35].

**Protein folding.** With amino acids contacting each other, the synthesised protein sequences fold into their functional three-dimensional structures. These three-dimensional structures are essential for the function of the proteins. They can be well-defined, as well as unstructured, completely or partially [36]. The latter situation is called the disordered states of proteins. According to *the Anfinsen's dogma* [37], protein three-dimensional structures are determined by amino acid sequences.

**Protein transport.** Protein transport refers to the process of transporting proteins to their destined locations [38]. It involves many signalling sequences, pathways and targeting processes so that the proteins are transported to the appropriate locations for the right purposes.

**Figure 1.1** demonstrates the process of gene expression including *transcription*, *post-transcriptional modification*, *translation*, *post-translation modification*, and *protein folding*. In this graph, the amino acid sequence [39] and the tertiary structure [40] of the protein *p53\_Human* was used.



**Figure 1.1** Gene expression with transcription, post-transcriptional modification, translation, post-translational modification, and protein folding.

As the primary product of gene expression, protein plays an essential role in the functioning of different organisms. It is 1) synthesised as amino acid sequences via

translation, possibly 2) coupled with post-translational modifications, and then 3) folded into its three-dimensional structure and 4) transported to its target locations. This four-step procedure of *protein expression* counts for half of the aforementioned steps in gene expression, which demonstrated the importance of protein analysis for the better understanding of the functional mechanism of life.

### 1.2.2 Protein sequences, structures and functions

For proteins, three inter-correlated concepts are vital for understanding the mechanism of protein functioning. They are *protein sequences*, *structures* and *functions*.

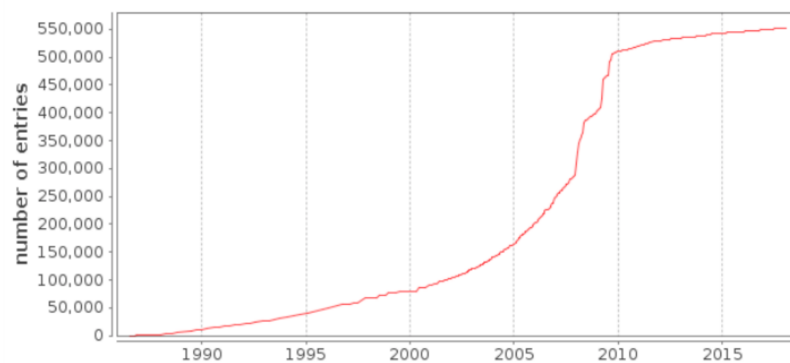
**Protein sequences.** *Protein sequences* define the primary structure of proteins [41]. They are linear sequences of amino acids that start from the N-terminal and end at the C-terminal. Each amino acid at a specific position in a protein sequence is referred to as an *amino-acid residue* (or *residue*). In most organisms, there are 20 naturally occurring amino acids, which are denoted using either the 1-letter notation system or the 3-letter notation system [42]. With the 1-letter denotation system, protein sequences are simply represented as strings of single alphabet letters, with sequence lengths ranging from 2 to ~35,000 [39].

**Protein structures.** It is more complicated to define protein structures. During the process of protein folding [43], amino acid sequences first form intermediate local segments of specific structural conformations, which later fold into their tertiary structures. The intermediate local segments are called the *protein secondary structures* (SS), and the two most common secondary structures are  $\alpha$ -helices and  $\beta$ -sheets [44]. The *tertiary structure* of proteins is the three-dimensional structure of the protein and is dependent on its primary and secondary structures [45]. It is defined according to the spatial arrangement of atoms in a protein. This spatial atom arrangement is referred to as *structural conformation*, which can change from one to another in the lifetime of a protein [46]. Among all the conformations, there is usually a dominant conformation that is the most stable, which determines the function of the protein and is called the native

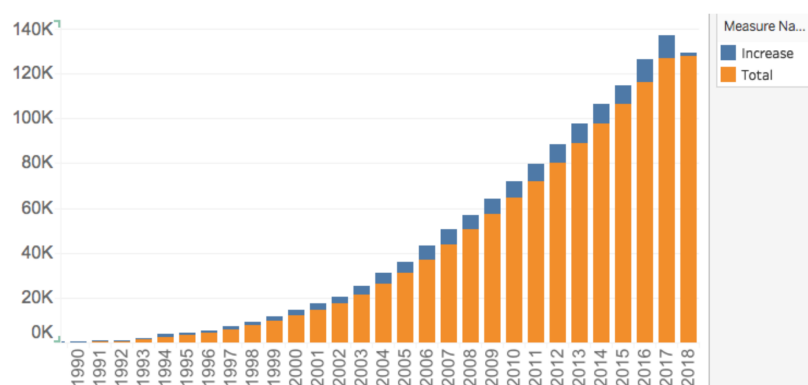
state [47]. Protein tertiary structure often refers to this native state. An exceptional case is the disordered or partially disordered proteins [48], for which there is a lack of native states.

**Protein functions.** Protein functions depend on the shape or conformation of the protein [49]. *Enzymes*, for example, are a specialized class of proteins responsible for catalysing chemical reactions within cells [50]. The structures of their active sites such as low barrier (unusually short) hydrogen bonds plays important roles in enzyme catalysis [51]. Another example is the *biological target*, which refers to proteins whose tertiary structure contain binding pockets for specific drug compounds [52].

With the availability of existing biotechnological techniques, it is more efficient and cheaper to identify protein sequences than to determine protein structures and functions. Up to the time of writing, according to the Swiss-Prot/UniProt database [39] - one of the most comprehensive databases of protein sequences - there are more than 90 million protein sequences being curated manually or automatically. These sequences are either inferred via biosynthesis process [53] or directly sequenced using protein sequencing technologies [54, 55]. In contrast, protein structures are determined using experimental methods such as *X-ray diffraction* [56] and *nuclear magnetic resonance (NMR)* [57] which are expensive and time-consuming to run. According to the RCSB PDB database [40], one of the most updated databases for protein structures, there are only ~125,000 proteins have their tertiary structures determined. **Figure 1.2** (a) and (b) respectively demonstrate the growth of the number of protein sequences and protein structures in the last few decades.



(a) The number of manually reviewed protein sequences in the Swiss-Prot/UniProt database.



(b) The number of identified protein structures in the RCSB PDB database.

**Figure 1.2** The increasing numbers of manually reviewed protein sequences in the Swiss-Prot/UniProt database and solved protein structures in the RCSB PDB database.

### 1.2.3 Protein structural and functional properties

Protein structures and functions represent two abstract concepts that are described using different sub-concepts such as protein secondary structures, intrinsic disorder, torsion angles and functionally important sites. In this thesis, we refer to each of these sub-concepts as a *protein structural/functional property*. In the UniProt/Swiss-Prot database [39], they are also referred to as *features*. We introduce some of the protein structural and functional properties as follows.

**Protein secondary structures.** *Protein secondary structures (SS)* refers to the intermediate state of local segments formed right after protein sequence biosynthesis



before further folding into three-dimensional structure<sup>2</sup>. According to the 3-class DSSP categorical system, each residue is assigned one of the three secondary structure elements, e.g. helix (H), strands (E) and coils (C) [58]. According to the 8-class DSSP categorical system, each residue is assigned one of the eight secondary structure elements including 3-turn helix (G), 4-turn helix (H), 5-turn helix (I), extended strand (E), isolated  $\beta$ -bridge (B), hydrogen bonded turn (T), bend (S) and coil (C) [58]. The secondary structure elements are annotated for each individual residue.

**Solvent accessibility.** As a complementary structural property to protein secondary structures, *solvent accessibility* is used to identify to what degree a residue is buried or exposed [59]. Various measurements are defined to represent solvent accessibility, including the solvent accessible surface area<sup>3</sup>, relative solvent accessible surface area<sup>4</sup>, contact numbers, residue depth and half-sphere exposure [61], among which the first two are usually used to quantitatively represent solvent accessibilities. Solvent accessibility is annotated at the residue-level.

**Backbone torsion angles.** *Backbone torsion angles* refer to the local structure in proteins [62]. Three types of torsion angles are defined, including  $\varphi$ ,  $\psi$  and  $\omega$ , among which the former two provide the flexibility required for the polypeptide backbone to adopt a certain fold, while the last one is essentially flat and fixed to 180 degrees [63].

**Residue contact map.** Protein residue-residue contact maps refer to pairs of residues between which the spatial Euclidean distance is smaller than 8Å [64]. Residue-residue contacts are labelled as short-range, medium-range and long-range depending on the sequential distance between two residues [64]. With residue-residue contacts, residues that are sequentially distant may be spatially close to each other, which, to

---

<sup>2</sup> Protein secondary structure: [https://en.wikipedia.org/wiki/Protein\\_secondary\\_structure](https://en.wikipedia.org/wiki/Protein_secondary_structure)

<sup>3</sup> The solvent accessible surface area (ASA) is defined as the locus of the centre of the solvent molecule as it rolls over the van der Waals surface of proteins. (60. Gromiha, M.M., *Chapter 3 - Protein Structure Analysis*, in *Protein Bioinformatics*. 2010, Academic Press: Singapore. p. 63-105.)

<sup>4</sup> The relative accessible surface area is defined to penalise amino acids that have larger maximum possible accessible surface area.

some extent, determines the tertiary structure of proteins. Different from protein secondary structure and solvent accessibility, residue contacts are annotated for pairs of residues.

**Protein intrinsic disorder.** Proteins and regions that form stable tertiary structures are referred to as the native structure while proteins and regions that do not or partially form stable tertiary structures are called the *intrinsically disordered proteins and regions* or *protein intrinsic disorder (IDP/IDR)* [48]. The discovery of disordered proteins and regions challenged the traditional protein structure-function paradigm that protein functions are determined by fixed protein tertiary structures [36]. While, in fact, disordered proteins and regions are highly abundant and their functional repertoire complements the functions of ordered proteins [65]. In practice, each residue in a protein is labelled as ordered or disordered, resulting in the sequential labelling of disordered states for protein sequences.

**Protein secondary structure population.** With disordered proteins and regions, protein sequences do not form stable tertiary structure, and correspondingly the secondary structure elements are dynamically assigned for each residue. Since it is not accurate to assign fixed secondary structure element to regions that do not form stable structures, *protein secondary structure populations (SSP)* were proposed to describe the quantitative probability distribution of secondary structure elements in disordered proteins and regions [66]. Similar to protein secondary structure, SSPs are labelled at the residue-level.

**Tertiary structures.** Protein tertiary structures are represented as the three-dimensional coordinates in  $x$ ,  $y$  and  $z$ -axis for each atom. They are currently determined by using the X-ray diffraction [56] and Nuclear magnetic resonance (NMR) [57] experimental methods. The Protein Data Bank (PDB) [40] defines the PDB file format<sup>5</sup>, which is widely used for representing the data that is derived from X-ray diffraction [56]

---

<sup>5</sup> PDB file format: [https://www.rcsb.org/pdb/static.do?p=file\\_formats/pdb/index.html](https://www.rcsb.org/pdb/static.do?p=file_formats/pdb/index.html)

and NMR [57] studies . Besides the three-dimensional coordinates, the PDB file format also describes other important structure-related information including experimental details (including pH values and temperatures), residue-residue contacts, and transformation between coordinate systems.

**Protein family.** *Protein families* refer to the groups of proteins that share the same evolutionary origin [67]. Proteins belong to the same protein family usually but not necessarily demonstrate sequential and structural similarities or related functions. Various protein family classification systems have been constructed based on sequential motifs [68], homology profiles [69, 70], and structural and functional similarities [71].

**Functional domain.** *Protein functional domains*, also known as *protein domains* or *conserved domains*, refer to parts of proteins that evolve, function and exist independently from the rest of the protein. Protein domains are considered the autonomous folding units [72], protein evolutionary modules [73], and functional units [74].

**Active site.** The *active site* of an enzyme (sometimes referred to as the catalytic centre) is that portion of the molecule that interacts with substrate and converts it into product [75]. It usually contains both the *binding sites* that forms enzyme-substrate complex via the lock-and-key model or the induced fit model [76], and the *catalytic sites* that catalyses the chemical reaction.

**Phosphorylation site.** *Phosphorylation* refers to a PTM that attaches phosphate-groups to the substrate proteins catalysed by a group of kinase called *protein kinases* [34]. The sites in substrate protein sequences to which the phosphate-group is attached are referred to as *phosphorylation sites*. Phosphorylation represents the most common PTM that plays essential roles in gene expression [35, 77, 78]..

The active sites, catalytic sites, binding sites, and phosphorylation sites are annotated at the residue-level and referred to as *functionally important sites (FISs)*. In total, Chakrabarti, S. and Lanczycki, C.J. (2007) summarised 16 FISs over 6 functional categories [79]. Please refer to **Appendix A** for the summarisation table for FISs.

**Drug-target interaction.** *Drug-target interaction (DTI)* refers to the specific interaction between drug compounds and their biological targets. Here, biological targets refer to molecules that are intrinsically associated with a particular disease process and that could be addressed by a drug to produce a desired therapeutic effect [80]. They are usually in the form of proteins or polypeptides whose tertiary structures form the docking pocket for drug compounds [52]. The identification of DTI represents a crucial step in drug development and drug repositioning [81].

**Protein-protein interaction.** *Protein-protein interaction (PPI)* refers to the physical contact with molecular docking between proteins that occur in a cell or in a living organism *in vivo* [82]. The identification of PPI is crucial for understanding biological functions because proteins rarely act alone. DTI and PPI are both annotated for pairs of molecules (e.g. a pair of a drug and a biological target, or a pair of proteins). They represent two of the most important interacting relationships in interactome, a complete map of the protein interactions that happen in living organisms [82].

**Table 1.1** summarises the structural and functional properties and their corresponding representations when modelled with computational methods.

In order to validate the effectiveness of the deep learning frameworks that are proposed in this thesis, we performed application analysis to the following five protein structural and functional properties, including *protein families*, *IDP/IDRs*, *SSPs*, *phosphorylation sites* and *DTIs*. Among these five structural and functional properties, protein families, as one of the most generic classification system of proteins, are annotated at the sequence-level; IDP/IDR and SSPs are structural properties that are annotated at the residue-level; phosphorylation sites represent the functionally important sites that are annotated at the residue-level; and DTIs are crucial for the understand of protein functions regulated by drugs at the biological interaction-level.

**Table 1.1** A summary of the protein structural and functional properties.

Property Name	Granularity	Category	Representations
<i>Structural properties</i>			
Secondary structure	Residue	Secondary	Helix, Strands, Coils (3-class)
Secondary structure population	Residue	Secondary	Populations of Helix, Strands, Coils
Solvent accessibility	Residue	Secondary	Accessible/non-accessible
Backbone and torsion angles	Residue	Secondary	Angles
Residue contact map	Residue interaction	Secondary	Contact $(r_i, r_j)$ where $r_i, r_j$ are two residues
Intrinsic disorder	Residue	Disorder	Ordered/disordered
Tertiary structure	Atom	Tertiary	Coordinates in x, y, and z-axis
<i>Functional properties</i>			
Functional domain	Residue	Domain	$(pos_{start}, pos_{end})$
Protein family	Sequence	Domain	Proteins that share the same evolutionary origin.
Active site	Residue	Site	Active/non-active
Phosphorylation site	Residue	Site	Phosphorylated/non-phosphorylated
Drug-target interaction	Biological interaction	Interaction	Interact $(p_i, d_j)$ where $p_i, d_j$ are proteins and drugs
Protein-protein interaction	Biological interaction	Interaction	Interact $(p_i, p_j)$ where $p_i, p_j$ are two proteins

#### 1.2.4 Databases for protein analysis

A number of databases are publicly available that provide annotations on and assist in the analysis of protein sequences, structures and functions. In this section, we introduce some of useful databases from which we collected data.

**The UniProt/Swiss-Prot database.** The Swiss-Prot/UniProt database [39] is one of the most complete collections of protein sequences. By the time of writing, the number of proteins has increased by 450% since 2012. It currently collected 555,594 manually reviewed protein and 90,050,711 unreviewed proteins [39]. The Swiss-Prot/UniProt database provides the main source of protein sequences based on which we predicted structural and functional properties. It also contains the most up-to-date protein annotations such as phosphorylation sites.

**The Pfam and InterPro databases.** Protein families are groups of proteins that share the same evolutionary origin [83]. However, there is not yet a consensus classification system for protein families. Existing protein family databases are constructed based on sequential or structural similarities or functional relations [70, 71, 84, 85]. To validate the performance of our proposed models, we collected annotations from the Pfam database [70] and the InterPro database [86]. The Pfam database curates a collection of protein families and functional domains that are generated using the HMMER programme [87]. The current version Pfam 31.0 records a total of 16,712 families and 604 clans that are cross-referenced to proteins or peptides in protein databases such as the Swiss-Prot/UniProt database. The InterPro database represents a meta database that integrates annotations from multiple protein family databases such as Pfam, PIRSF [84], and CATH-Gene3D [88].

**The DisProt and MobiDB databases.** The DisProt database [89] is a bioinformatics data source that contains manually curated IDP/IDRs. The current version DisProt 7 v0.5 contains 803 and 2,167 IDPs and IDRs respectively. A more recent database MobiDB

for protein intrinsic disorder was release in 2012 [90], 2014 [91] and 2018 [92], which contains annotations for both IDP/IDRs and SSPs that are generated by both manual curation and computational predictions.

**The BMRB database.** The BMRB database [93] collects, annotates, archives, and disseminates the important spectral and quantitative data derived from NMR spectroscopic investigations of biological macromolecules and metabolites<sup>6</sup>. For protein SSP prediction, we extracted 12,018 BMR entries based on which we calculated the secondary structure populations for validating proposed predictive models.

**The Phospho.ELM database.** The Phospho.ELM database [94] collects phosphorylation sites for different protein kinases in eukaryotes. It contains phosphorylation site annotations for 8,718 substrate proteins. However, this database stopped updating in 2011. Therefore, to enlarge our datasets, we also extracted and combined the up-to-date phosphorylation site annotations from the Swiss-Prot/UniProt database [39].

**The KEGG database.** The KEGG database is a comprehensive data source that stores drugs, pathways, compounds, genes and proteins, genomes, enzymes, reactions, and diseases [95, 96]. In this thesis, we used training and testing datasets that were constructed from the KEGG database for DTI prediction.

Besides the above eight databases, we also employed data from various other databases such as the RCSB PDB database [40], the DrugBank database [97], and the PhosPhAt database [98].

### 1.2.5 Bioinformatics tools for protein analysis

The analysis of protein sequence, structures and functions strongly rely on the existing bioinformatics tools. In this thesis, we employ the following tools for the purpose of data collection and feature calculation.

---

<sup>6</sup> The BMRB database: <http://www.bmrb.wisc.edu/bmrb/>

**NCBI-BLAST.** NCBI-BLAST [99] is a tool suite that performs biological sequence alignment. It operates by a heuristic method where short matches between biological sequences are seeded for exploring longer matches in multiple iterations. In various predictive scenarios, we used its specialised tool PSI-BLAST [100] to generate *the position-specific scoring matrix (PSSM)*, which is widely used as a homology profile encoding the evolutionary information of biological sequences.

**DISOPRED3 and PSIPRED.** DISOPRED3 [101, 102] and PSIPRED [103, 104] are computational tools for predicting protein intrinsic disorder and secondary structures from protein sequences. For predicting protein structural or functional properties that are related to these two protein structural properties, we used DISOPRED3 and PSIPRED to generate the corresponding structural information.

**The  $\delta 2D$  method.** The  $\delta 2D$  method [66] was proposed to calculate protein SSPs from NMR chemical shifts. Since protein SSP prediction represents a relatively new prediction task, there is a lack of manually curated databases and a limited number of tools for generating such information. Therefore, we generated the annotation data for protein SSPs using the  $\delta 2D$  method only.

Apart from the above bioinformatics tools, we also implemented local source code packages for collecting and extracting data and generating protein sequence-based information.

## 1.3 An introduction to the background of deep learning

In this section, we introduce deep learning with respect to its history, categorization, and the advantages and disadvantages of different deep learning models.

### 1.3.1 The history and development of deep learning

Deep learning refers to the multi-layer architectures for learning data representation with multiple layers of abstraction [105]. It was first applied in areas such as natural language processing [12], image processing [106], and speech recognition [13]. Despite the more



recent attention drawn to deep learning, the idea of using multiple layers of perceptrons to model complex problems started in early 1965 [107].

For a long time, the development of multi-layer neural networks was relatively slow due to multiple issues including slow training processes and a lack of data. The slow process of training was considered to be related to many factors, such as the selection of activation functions, the initialisation of parameters, and the vanishing gradients [108, 109]. The situation only got improved after 2006, when Prof. Geoffrey Hinton proposed to learn the data representation one layer after another, treating the training of each layer as an unsupervised learning process [110]. Since then, new deep learning techniques such as autoencoder [111] and dropout [112] have been proposed to address the training issues of deep learning models. At the same time, the fast development of deep learning was also boosted by the rapid development of computational power [113] and the massive amount of data available on the Internet.

### 1.3.2 An introduction to different deep learning models

The fundamental idea of deep learning is rather straightforward - if the data cannot be represented with shallow models [3], for example, single-layer neural networks - models with multiple hidden layers should be capable of modelling the complex data. The earliest deep neural networks were *fully-connected* where each neuron in one hidden layer is connected to every neuron in the previous layer [114]. They were later designed to involve less weights and function better for specific tasks. We introduce some of the deep learning models as follows.

**Convolutional neural networks.** The *convolutional neural network (CNN)* is the most widely used deep learning models that has been applied in different areas. Its design was inspired by the receptive field of the cat visual cortex [115]. Different from fully-connected neural networks, in CNN, each neuron in one hidden layer is connected only to a restricted number of neighbouring neurons in its adjacent layers [116]. It encourages the modelling of local correlations among data. Examples of such local

correlations can be found in images where pixels have higher correlations to their neighbouring pixels and in sentences where words have higher correlations to their nearby words. Further, such restrictions decreased the number of parameters, resulting in models requiring less training data and training time.

**Recurrent neural networks.** The *recurrent neural network (RNN)* was designed specifically for sequential data such as natural languages and speech signals [117]. It defines a temporal model where each step corresponds a single element in the input sequence. By introducing a state signal that is passed on from one step to another and an output signal which determines the label of individual elements, the history information encoded in the previous steps is retained and passed on to the later steps [118]. The advantage of the RNN model is that the history information from previous (or later) steps can be maintained and used for predicting information of the current step.

**Generative adversarial network.** The *generative adversarial network (GAN)* was recently proposed to estimate generative models via an adversarial process [119]. It is composed of a generative module and a discriminative module. The former is optimised to capture the true distribution of the data, while the latter is optimised to discriminate the samples generated from the former and those that are drawn from the true distribution of the data [119]. The learning process is designed as a sum-product game between these two modules. The GAN model has been applied to synthesise images from text descriptions [120].

Other deep learning models include *the Boltzmann machines* [121], *the Deep Belief Nets* [122], and *the Sum-Product networks* [123]. However, the above three deep learning models (especially the first two) are the most widely used in biological sequence analysis. Many variations of the first two models have been applied to address the prediction tasks in protein analysis. Detailed discussions are provided in **Section 2.1.2**.

### 1.3.3 Advantages and disadvantages of deep learning

Deep learning not only refers to a collection of machine learning models, but it also refers to new designing mindsets that can be used to address the challenges that most machine learning methods are facing. More specifically, it has the following several traits that may help improve the prediction performance of protein structural and functional properties.

Firstly, deep learning enables an effective implementation of *transfer learning* and *multitask learning*. Transfer learning or *inductive transfer* was proposed in the early 1990s to apply the knowledge learned from one domain to another domain [124]. It is closely related to multitask learning where multiple tasks are solved together so that the training signals of related tasks can be used to improve the performance of individual tasks [125]. Transfer learning and multitask learning explores the similarities and connections between the source task and the target task or related tasks, which has been used as a strategy to address the issue of a lack of training data. With deep learning, pre-trained hidden layers of the source task are regarded as representation extractors [126], which can be used subsequently by other related tasks as the knowledge transferred from the source task. Alternatively, the hidden layers of related tasks can be trained together to generate representations that are supportive for each other, thereby facilitate the purpose of multitask learning.

Secondly, deep learning models have the prediction time complexity linear to the number of their weights. Despite the relatively long time required for training deep learning models, the prediction time of a pre-trained deep learning model is relatively short, depending on the number of weights in the model. Compared to models which require real-time calculation and approximation, such as probabilistic graphical models [127] and K-nearest neighbours [128], deep learning models provide fast execution in prediction time.

Thirdly, deep learning learns multiple layers of representations [110]. Data representation is extracted in multiple abstractive layers in deep learning, with the

purpose of discovering more abstract features from higher levels of the representation [3]. In applications like facial recognition, based on the raw pixel inputs, Krizhevsky, A., Sutskever, I. and Hinton, G.E. (2012) learned the frequency- and orientation-kernels and coloured blobs using the first convolutional hidden layer [5], and Huang, G.B., Lee, H. and Learned-Miller, E. (2012) learned facial regions using the second convolutional hidden layer. Here, the raw pixel inputs, the kernels and coloured blobs, and the facial regions represent three different levels of abstract features, from low to high. In most cases, higher abstract features are easier to discriminate.

Since deep learning models are mostly implemented as multilayered neural networks, they derived some of the disadvantages that are shared by neural network models. Firstly, despite the continuous effort made by the community for explainable artificial intelligence, it is still a challenge to understand how deep learning models come to a decision [129]. Secondly, the training of deep learning models requires a significant amount of training data, which limited the application of deep learning in many areas where there is a lack of labelled data. Thirdly, the training of deep learning models is time-consuming [130], despite its low time complexity in prediction. Finally, existing training of large deep learning models relies on the usage of high performing computation GPUs that are designed specifically for deep learning training [5], such as GPU Tesla and TPU cloud. Such resources are less accessible for small industrial organisations or academic teams to train their own deep learning models.

## **1.4 Existing challenges, research questions and objectives**

In this section, we propose the following three research questions based on existing challenges of applying deep learning techniques to protein analysis. For each of the research questions, we also raised the corresponding objectives to be achieved in this thesis.

### 1.4.1 Research question #1

*How to effectively represent protein sequences for different granularities of prediction tasks and, specifically, how do purely data-driven representations perform compared with biological heuristic-based representations?*

To predict protein structural and functional properties directly from protein sequences, the first crucial step is to represent protein sequences as effective input features. On one hand, existing protein sequence representations, such as homology profiles, physicochemical properties and structural information, are generated based on biological heuristics, which ignore the abundant information that is encoded in large amounts of raw protein sequences. Therefore, it is interesting to explore the generation of representations that are purely driven by protein sequence data and compare their performance to biological heuristic-based representations as the input feature in protein sequence-based prediction tasks.

On the other hand, since protein sequences are of different lengths, existing representations are either designed for sequence-level predictions or residue-level predictions. There is a lack of condensed vector representation that can be applied to different granularities of prediction tasks. According to **Table 1.1**, protein family and domain annotations are labelled at the sequence-level; structure annotations and functionally important site annotations are labelled at the residue-level; while drug-target interactions and protein-protein interactions are labelled at the biological interaction-level. Therefore, it is important to develop a protein sequence representation that can be applied to three different granularities of prediction tasks.

The objective of this research question is to explore the different strategies to generate vector representations from the large amount of protein sequence data, and systematically evaluate its performance in three different granularities of prediction tasks.

### 1.4.2 Research question #2

*How to design a deep learning framework that can be applied to predict different protein structural and functional properties?*

According to **Section 1.2**, protein structures and functions are described as multiple specific properties that can be quantified in predictive modelling. Especially for protein structures, dozens of structural properties are used to provide detailed descriptions of the different aspects of a protein structure. However, from the perspective of deep learning, the prediction of all structural properties at the same granularity can be generalised as a single prediction task, and the predictive model of a specific structural property can be trained based on specific training data. For example, the prediction of sequence-level protein structural properties can be summarised as follows,

1. It takes the representation of the whole protein sequence as the input;
2. It produces an output for each input protein sequence;

With the input and output generalised from a group of prediction tasks, deep learning frameworks can be designed at a more abstract level, which allows the flexibility of the selection of input features, the design of the structure of the deep learning models, and the modelling of the expected prediction outputs.

The objective of this research question is to design a deep learning framework that can be applied to predict any individual protein structural or functional property.

### 1.4.3 Research question #3

*How to leverage limited quantities of labelled biological data for rigorous model training by incorporating biological background knowledge?*

Compared to fields such as image processing, language processing and speech recognition, it is relatively more expensive and challenging to obtain labelled data from biological experiments. Especially when it comes to specific tasks such as protein structure prediction, the labelled data is derived from X-ray and NMR experiments which

are expensive and labour-intensive to conduct. In addition, there is a lack of developed data-sharing mechanisms in biology, due to the issues of intellectual property, funding restriction, and ethics.

At the same time, there is a lack of a global view to bridge the biological and computational aspects during the process of model design. Most of the existing machine learning methods are applied to predict protein structures and functions in a task-specific manner, which makes the predictions mutually isolated from each other. Considering the abundant background knowledge of biology, it is crucial to explore the inter-correlation between different aspects of the same biological object (e.g. a specific protein or residue), which in turn can affect the design of the predictive deep learning frameworks.

The objective for this research question is to design deep learning frameworks by incorporating biological background knowledge so that predictive models can be rigorously trained even with limited quantities of labelled biological data.

## 1.5 Contributions and scope

To address the three research questions raised in the previous section, we design different algorithms, models and frameworks; introduce different benchmark datasets and evaluation systems; and conduct extensive benchmarking experiments and case studies. We summarise the main contributions of this thesis as follows.

*We propose deep learning solutions for each of the three research questions.* Firstly, we proposed new implementations of the distributed representation of protein sequences and their applications in three granularities of prediction tasks, including whole sequence-level predictions, residue-level predictions and biological interaction-level predictions. Secondly, we propose the general deep learning framework, *DeepS(sequence)2P(property)*, for predicting protein structural and functional properties based on protein sequences. Thirdly, based on the qualitative correlations between protein structural properties, we propose the multitask deep learning frameworks, *Multi-task* and *Cross-stitch*, for predicting multiple protein structural properties simultaneously.

Fourthly, inspired by the hierarchical classification systems of protein, enzymes, and protein kinases, we propose the deep transfer learning framework, *DeepTransfer*, for predicting functionally important sites of enzymes that are pre-organised in hierarchical structures.

*In order to validate the effectiveness of the proposed frameworks, we apply them to five specific prediction tasks.* More specifically, we apply the distributed representation *prot2vec* to sequence-level protein family prediction, residue-level phosphorylation site prediction, and biological interaction-level DTI prediction. Based on the DeepS2P frameworks, we perform systematic evaluation of the resulting predictive models for structural properties such as IDP/IDRs and SSPs and functional properties such as phosphorylation sites. As for the multitask frameworks, we take into consideration the observed correlations between IDP/IDRs and SSPs and evaluate the resulting multitask deep learning models for predicting IDP/IDRs and SSPs simultaneously. Finally, with the existing classification of protein kinases in the hierarchy of groups, families and subfamilies, we apply the DeepTransfer to phosphorylation site prediction with the purpose of improving prediction performance for kinases with limited labelled phosphosites.

*We perform critical analysis of the performance of the proposed frameworks and generate publicly available bioinformatics tools with user-friendly interface and promising performance.* For each of the prediction tasks, we perform systematically evaluate the prediction performance of the proposed deep learning frameworks. For those that competed less favourably with existing methods, we conduct quantitative investigations to analyse the factors that may have affected the prediction performance. For those with promising results, we develop bioinformatics tools that are publicly available to other researchers. We then summarise a set of heuristics for designing deep learning frameworks for biological data analysis.

## **1.6 An overview of the thesis structure**



This thesis is organised into nine chapters. **Figure 1.3** illustrates the overall structure of this thesis, indicating the internal relationships among the following chapters and their enclosed sections.

**Chapter 2** gives an overview of the prerequisite backgrounds, fundamental algorithms, models and methods, and respective methods to which we compare the proposed models in different prediction tasks.

**Chapter 3** introduces the scientific design for applying machine learning methods to protein analysis. The scientific design demonstrates a general procedure of data collection, feature construction, model learning, and performance evaluation, which is conducted for each of the more specific tasks that are addressed in this thesis.

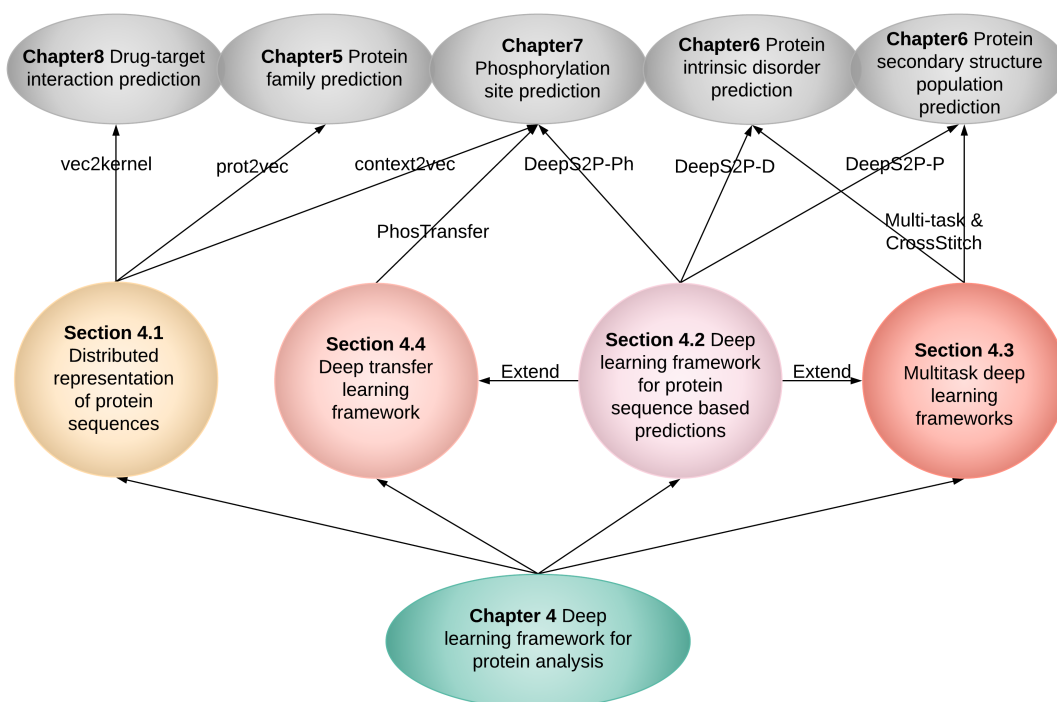
**Chapter 4** presents the deep learning techniques and frameworks that are proposed for addressing each of the research questions in protein analysis. Firstly, it introduces the distributed representation of protein sequences and its applications in whole sequence-level (*prot2vec*), residue-level (*context2vec*), and biological interaction-level (*vec2kernel*) predictions. Secondly, it proposes a protein sequence-based deep learning framework, *DeepS2P*, for predicting any individual residue-level protein structural /functional property. Thirdly, it proposes the simultaneous prediction of two or more protein structural properties using multitask deep learning frameworks. Finally, it introduces the application of the deep transfer learning framework, *DeepTransfer*, to the prediction of functionally important sites whose catalytic enzymes are classified in hierarchical structures.

In order to demonstrate the effectiveness of the proposed deep learning frameworks, **Chapter 5 – Chapter 8** introduces the applications of the proposed frameworks in five prediction tasks, including

- the prediction of protein families;
- the prediction of phosphorylation sites;
- the prediction of protein intrinsic disorder (IDP/IDRs);

- the prediction of protein secondary structure populations (SSPs), and
- the prediction of drug-target interactions (DTIs).

Since the predictions of IDP/IDRs and protein SSPs are performed simultaneously using the multitask frameworks, these two prediction tasks are presented together in **Chapter 6**.



**Figure 1.3** The thesis structures with respect to chapters and sections.

**Chapter 5** introduces the prediction of protein families using the prot2vec representation. In this chapter, the prediction performance of prot2vec as an input feature vector is systematically evaluated in protein family prediction. In particular, as a purely data-driven representation, prot2vec's prediction performance in protein family prediction is compared with that of the homology profile-based features and physicochemical property-based features that are generated with biological heuristics.

**Chapter 6** introduces the prediction of IDP/IDRs and SSPs using the DeepS2P framework respectively and using the multitask frameworks simultaneously. In this

chapter, DeepS2P-based deep learning models are compared with shallow architectures in protein structural property prediction. Further, the DeepS2P-based single-task predictive models are compared with multitask predictive models, demonstrating the effects of employing related prediction tasks as the supporting task in multitask deep learning frameworks.

**Chapter 7** introduces the prediction of phosphorylation sites using the context2vec representation, the DeepS2P framework, and the DeepTransfer framework, respectively. Firstly, the distributed representation, context2vec, is evaluated as a complementary input feature to biological heuristic-based features in phosphorylation site prediction. Secondly, DeepS2P-based predictive model is compared with the DeepTransfer-based predictive model, for predicting phosphorylation site of kinases with limited phosphosite annotations, illustrating the importance of designing deep learning frameworks by incorporating biological background knowledge, especially when there is a lack of annotated data.

**Chapter 8** introduces the prediction of drug-target interactions using the vec2kernel representation as the protein kernel. In this chapter, comprehensive comparison experiments are conducted to evaluate the performance of vec2kernel compared with that of the existing protein kernels, under different experimental settings.

**Chapter 9** summarises the work of this thesis by concluding major contributions and potential future work.

## 2 Literature Review

The application of deep learning in protein analysis relies on the avalanche of related works from both the machine learning area and the bioinformatics area. On one hand, the development of deep learning techniques demonstrated the elegant applications of machine learning to various application areas, providing analogous examples for extending the application of deep learning to the analysis of protein sequence, structures and functions. On the other hand, existing methods for protein analysis provide mature methodologies based on which we conduct corresponding research works and baseline predictive models to which we compare the proposed deep learning frameworks.

In this chapter, we perform the literature review in three steps. **Section 2.1** provides a technical overview of deep learning techniques. **Section 2.2** reviews the existing application of deep learning techniques in protein analysis. Finally, in **Section 2.3**, we present existing computational methods that are developed for predicting protein structural and functional properties.

### 2.1 Deep learning models, strategies and techniques

#### 2.1.1 Basics of machine learning and neural networks

Machine Learning is the field of scientific study that concentrates on induction algorithms and on other algorithms that learn from observations and make predictions [131]. It was first proposed by Arthur Samuel in 1959 [132], and has undergone substantial development together with some other fields including pattern recognition [133], artificial intelligent [134], data mining [135], and statistics [135]. Machine learning and pattern recognition can be regarded as the two facets of the same field, where the former was developed from computer science while the latter has its engineering origin [133]. Machine learning is often considered as an application or an approach of achieving

artificial intelligent [134]. It is also considered to be at the core of data mining since the key step of pattern discovery in data mining can be addressed by most machine learning algorithms. Yet, machine learning and data mining have different concentrations. The former focuses on the learning and induction step which aims to construct models reflecting the underlying structure behind the data, whereas the latter focuses on the discover of new knowledge [135]. Finally, according to Michael I. Jordan, machine learning has a long prehistory in statistics [136].

Machine learning has been widely applied to address multiple types of tasks, including classification, regression, and clustering.

**Classification.** A typical classification task seeks to classify unseen examples to two or more pre-defined classes. With machine learning, classification models are learned from a set of <example, class label> pairs, which are later used to assign class labels to unseen examples. Most of the predictive tasks in protein sequential, structural and functional analysis can be modelled as classification tasks. For example, the prediction of protein secondary structures classifies each target residue as one of the three protein secondary structure elements including helix, strands and coil [137-139].

**Regression.** The regression task seeks to estimate the correlation between the dependent variable and a set of independent variables [140]. When addressed using machine learning, the solution is similar to that of the classification task, except that the output is continuous instead of discrete. In protein analysis, protein backbone torsion angle prediction has been modelled as a regression task [141].

**Clustering.** Clustering seeks to divide a set of examples into groups according to a pre-defined distance measurement. However, there is no pre-defined classes. Therefore, the number of resulting groups varies depending on the distance measurement that is selected. Protein sequence clustering represents a typical example of the clustering task [142]. It divides a set of protein sequences into groups according to the distance among the protein sequences. The distance measurement can be selected as the edit distance or homologous distance.

Machine learning is generally categorised as either supervise learning or unsupervised learning. Supervised learning refers to techniques used to learn the relationship between independent variables and a designated dependent variable [131]. Take classification as an example, in supervised learning, data labelled with classes are used as the observation to guide the training of the model. In comparison, unsupervised learning refers to learning techniques that group instances without a pre-specified dependent variable [131]. As a result, there is no pre-defined labels and the task itself also involves the discovery of labels or classes. In practice, it is expensive and labour-intensive to obtain the labelled data, which leads to the third type of machine learning named semi-supervised learning [143]. Typically, in semi-supervised learning, labelled data and unlabelled data are combined to guide the process of model training.

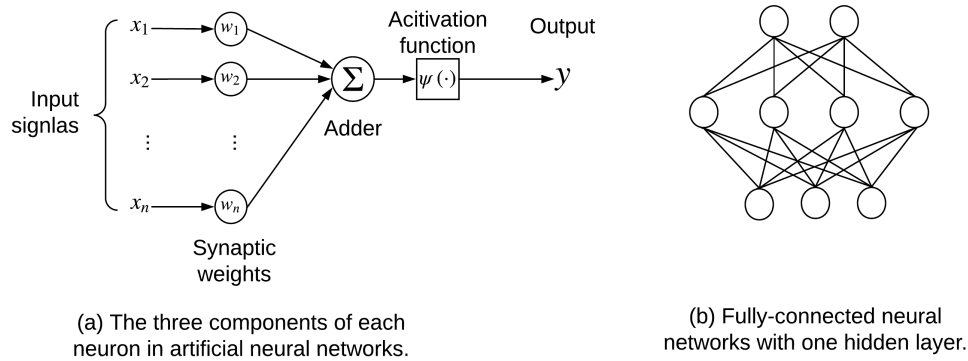
In the broad field of machine learning, multiple approaches were proposed and developed in the last fifty years, including artificial neural networks [144], support vector machines [145], probabilistic graphical models (including Bayesian networks) [127], decision trees [146], inductive logic programming [147], genetic algorithm [148], reinforcement learning [149] and clustering algorithms [150]. Some of these approaches are more advantageous to address one or two of the three tasks mentioned above. Decision trees, as an example, are specifically designed to address classification tasks. While the clustering algorithms are specifically designed to address clustering tasks.

Deep learning is often (but not necessarily) classified as a kind of neural network which is defined as follows [151],

**Definition 2.1** A neural network is a massively parallel distributed processor made up of simple processing units, which has a natural propensity for storing experiential knowledge and making it available for use. It resembles the brain in two respects:

1. *Knowledge is acquired by the network from its environmental through a learning process.*
2. *Interneuron connection strengths, known as synaptic weights, are used to store the acquired knowledge.*

It was inspired by biological neural networks where neurons are connected as a complex network to process information. Each neuron in artificial neural networks have three basic components, including the synaptic weights, the adder and the activation function [152]. **Figure 2.1** (a) demonstrates the basic architecture of a neuron.



**Figure 2.1** The architecture of a basic neuron in artificial neural networks.

For each neuron, the adder linearly combines the weighted inputs to general the weighted sum of the input signals, while the activation function allows the non-linear modelling of the relationship between input signals and the output. In practice, multiple neurons are connected to process the input information. **Figure 2.1** (b) demonstrates an example of the connectivity among multiple neurons. In this neural network, the nine neurons are organised in three layers, where the neurons in the same layer are not connected to each other but connected to every neuron in its adjacent layers. Neural networks with this particular organization of neurons and connectivity are referred to as the fully-connected neural networks [114].

Neural networks have been applied to many different protein analysis tasks, including disordered protein prediction [153], phosphorylation site prediction [35, 154], and protein secondary structure prediction [155]. In general, the input information for each target residue is first encoded to the input signals (or feature representations)  $x_1, x_2, \dots, x_n$ , and

its corresponding labels (e.g. disordered/ordered, phosphorylated/non-phosphorylated, helix/strands/coil) are used as the output  $y$ . With sufficient observations of pairs of  $\langle x_1, x_2, \dots, x_n, y \rangle$ , the learning algorithm is expected to assign values to the synaptic weights  $w_1, w_2, \dots, w_n$ , so that the resulting neural network is capable of generating the output  $y$  for unseen target residues.

### 2.1.2 Deep learning models

Deep learning is developed based on multi-layered neural networks. Depending on the restriction of connectivity and the different organization of neurons and connectivity, the most widely used deep learning models include convolutional neural networks [116], recurrent neural networks [118], and generative adversarial networks [119]. Here, we introduce convolutional neural networks, recurrent neural networks and the convolutional neural fields, which have been applied to the prediction of protein structures and functions.

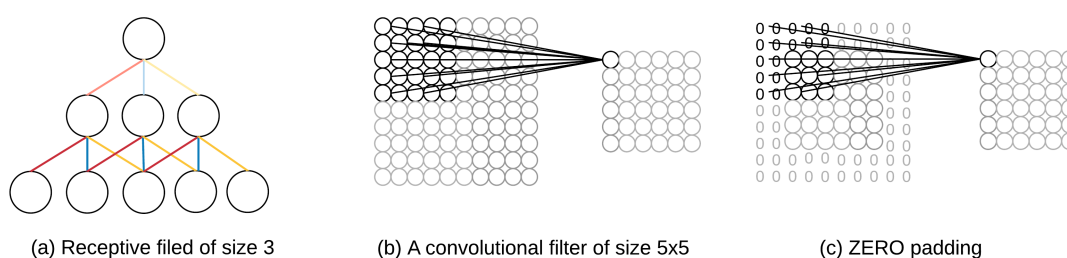
**Convolutional neural network.** As mentioned in **Section 1.3.2**, the convolutional neural network (CNN) was inspired by the animal visualization cortex where each neuron is only activated by a limited region in the visual field which is called the *receptive field* [115]. Instead of connecting to every neuron in the previous layer  $m - 1$ , each neuron in the current layer  $m$  is connected to a limited number of adjacent neurons in layer  $m - 1$ . With this restriction, the coverage of neurons in lower layers overlap with each other and therefore form a convolutional architecture. **Figure 2.2** (a) demonstrates the artificial neural network simulation of the receptive field of size 3.

In convolutional neural networks, the receptive field is implemented by using the convolutional filter. It defines how many neighbouring neurons in the previous layer are connected to the target neuron in the current layer [5]. In two-dimensional data, e.g. pixels in images, the convolutional filters are usually defined to have two dimensions. For example, a convolutional filter of size  $5 \times 5$  defines that five neurons in each of  $x$  and



y-axis from the previous layer convolve to one neuron in the current layer, as is shown in **Figure 2.2** (b).

When applying convolutional filters, padding strategy determines how to deal with elements (e.g. pixels, words, residues) at the border of the representation. If the *SAME padding* is applied in a 5x5 convolutional filter, two additional rows/columns of the same border values are added to compute the output for border elements. The *ZERO padding* sets the additional rows/columns with value 0, as shown in **Figure 2.2** (c).



**Figure 2.2** Receptive field, convolutional field and ZERO padding in convolutional neural networks.

The advantage of the CNN model is that it models local correlation within data in a hierarchical way. The restrictions in receptive fields enable the mapping between hidden layers and abstract layers in data. For example, in facial recognition, the input hidden layer is mapped to the input pixels, the first hidden layer is mapped to geometry patterns such as lines and angles, the second hidden layer is mapped to facial parts such as noses and eyes, the third hidden layer is mapped to super areas consisting of multiple facial parts, and the last layer is mapped to the whole face.

When applied to protein sequential modelling, the convolutional neural networks are mainly used to model the local correlation among adjacent amino acids.

**Convolutional neural fields.** The model of the deep convolutional neural fields (CNF) was proposed to estimate the image depth where the depth values are continuous and locally correlated to each other [156]. Compared to the CNN model, the CNF model

not only models the local correlation within feature space, it also enforces the modelling of local correlation in predicted data.

Deep convolutional neural fields consist of a CNN component and a conditional random fields (CRF) component. Let  $x$  be the input feature, and  $y = (y_1, y_2, \dots, y_N)$  the continuous labels to be predicted for position 1 to  $N$ . The conditional probability distribution of the data in the CRF component can be modelled as,

$$P(y|x) = \frac{1}{Z(x)} \exp(-E(y, x)) \quad [\text{Eq. 2.1}]$$

where  $E$  is the energy function and  $Z$  the partition function [127]. To model both the individual elementary data and the local correlation among continuous elements, *the energy function  $E(y, x)$  was split into a unary function  $U(y_p, x)$  and a pairwise function  $V(y_p, y_q, x)$* . The output of the unary function comes from a separate CNN for each individual element; while the output of the pairwise function was calculated from each pair of neighbouring elements with multiple types of similarities. The combined functions are later fed into a CNF loss layer which generates the loss values and guides the training of the whole network. In image depth estimation, the three similarities used were the colour difference, colour histogram difference and pattern disparity in terms of *local binary patterns* (LBP) [156].

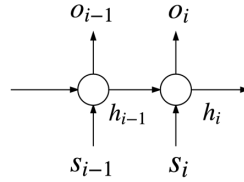
The deep CNF model has its advantage in modelling sequential data with sequential annotations where the local correlation among the sequential annotations would make a difference in prediction performance. It has been applied to predict protein secondary structures [157] and intrinsic disorder [158], where there are local correlations among the predicted neighbouring secondary structures and the predicted neighbouring disordered states.

**Recurrent neural network.** Recurrent neural network (RNN) was proposed to solve sequential problems. Different from the CNN model and the deep CNF model in which only the correlation among neighbouring elements is modelled, in the RNN model, there

is a component called the hidden state that ‘remembers’ the history information from previous positions.

Suppose the sequential input data is formalised by a sequence of symbol  $S = s_1, s_2, \dots, s_N$  where each element is represented by  $s_i$  and the output sequence is represented by  $O = o_1, o_2, \dots, o_N$ . Due to the various lengths of sequences, the units in the recurrent network are defined in a recurrent way, where the output at position  $i$ , *i.e.*  $o_i$ , is computed from  $s_i$  and  $s_{i-1}$  which can be generalised to all possible positions.

**Figure 2.3** demonstrates the general definition of a unit in the recurrent network.



**Figure 2.3** Consecutive units in recurrent neural networks.

Here, the output  $o_i$  is computed as follows according to *the Elman network* [159],

$$h_i = \sigma_h(W_{sh}s_i + W_{hh}h_{i-1} + b_h)$$

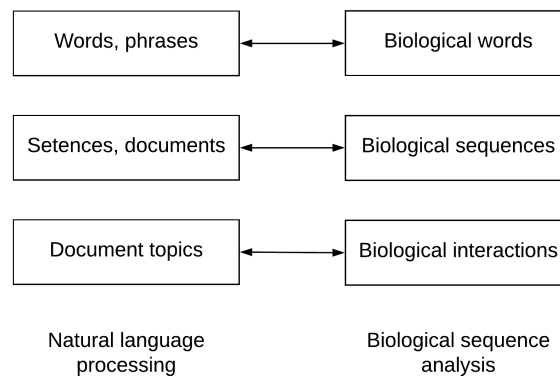
$$o_i = \sigma_o(W_{ho}h_i + b_o) \quad [\text{Eq. 2.2}]$$

where  $h_i$  is the hidden units at position  $i$ ,  $\sigma_h$  and  $\sigma_o$  are activation functions,  $W_{sh}$ ,  $W_{hh}$  and  $W_{ho}$  are weights and  $b_h$  and  $b_o$  are biases. The hidden states  $h_i$  works as the memory from previous positions. Commonly used variations of RNN include *the long short term memory (LSTM) network* [117], the bidirectional RNN [160], the gated RNN [161], and the hierarchical RNN [162].

A bidirectional LSTM model has been applied to predict disordered proteins and regions [153]. In this model, residues around the target position  $i$  are modelled step-by-step using two layers of LSTM units so that the long-distance dependencies can be used for the prediction of disordered states.

### 2.1.3 Distributed representation of words, sentences and documents

Ganapathiraju, M., et al. (2005) proposed the mapping between biological sequences and natural languages, where gene/protein sequences were mapped to raw texts from documents; protein expression, structures and functions were mapped to sentence semantics; and biological interactions were mapped to the topics of the documents [163] (**Figure 2.4**). Therefore, the modelling of natural languages provides analogous ideas for the modelling of protein sequences.



**Figure 2.4** The mappings between natural language processing and biological sequence analysis.

In natural language processing, the technique for mapping words and phrases to their vector representations is called word embedding [164]. It was originally implemented with counting-based techniques such as Term frequency (TF) [165] and term frequency-inverse document frequency (TF-IDF) [165] and *localist representation* techniques such as the 1-of-K encoding (*i.e.* one-hot encoding) [133]. A brief introduction of TF, TF-IDF and 1-of-K coding is as follows,

**Term frequency (TF).** The TF technique represents each sentence or document as a vector of the counts of each word in the dictionary. For example, for document  $D$  the TF representation  $(c_1, c_2, \dots, c_n)_D$  for vocabulary words  $(w_1, w_2, \dots, w_n)$  indicates the number of word  $w_1, w_2, \dots$ , and  $w_n$  in the document  $D$ .

**Term frequency - inverse document frequency (TF-IDF).** The TF-IDF technique was proposed to penalise words that are statistically more likely to appear in a language, such as 'a', 'the', and 'is'.

Both of the TF and TF-IDF techniques are based on the bag-of-words model [166] in which documents are represented as a bag of its words, disregarding the order of the words or any language grammars.

**1-of-K encoding.** In the 1-of-K encoding, each position in sentences or documents is represented as a vector of the size of the vocabulary, in which each element corresponds to a vocabulary word and its corresponding value indicates whether it appears in the current position. Therefore, the sentence or document itself is represented as a vector of vectors, *i.e.* a matrix. Such 1-of-K encoding matrix is considered as an implementation of the localist representation, which is defined as follows,

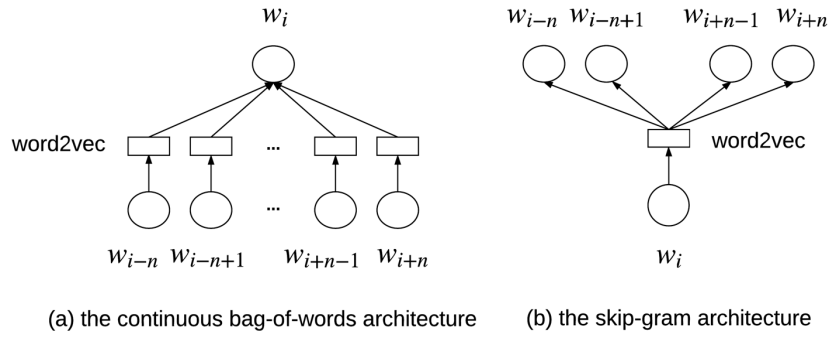
**Definition 2.2** In the localist representation, each semantically discrete item, concept or idea is associated with the activity of a single node [167].

In comparison to the localist representation, the distributed representation was first proposed by Hinton allow representing multiple items (, concepts, or ideas) with multiple nodes. The original definition is as follows,

**Definition 2.3** Each entity is represented by a pattern of activity distributed over many computing elements, and each computing element is involved in representing many different entities [168].

Recently, the distributed representation of words, phrases and documents [169, 170] was proposed based on the semantic correlation between words. These distributed representations are automatically generated from hidden layers of the generative models that are trained to generate words in natural language sentences. The distributed representation of words and phrases come in two flavours, *i.e. the continuous bag-of-*

words (CBOW) [171] architecture and the skip-gram [170] architecture. Both of these two architectures are constructed based on the feedforward neural network language model (NNLM) which explores the correlation between the current word  $w_i$  and its neighbouring words  $w_{i-n}, w_{i-n+1}, \dots, w_{i+n-1}, w_{i+n}$  (where  $n$  represents the maximum distance between the target word and its surrounding words). **Figure 2.5** (a) and (b) demonstrate the two architectures, respectively.



**Figure 2.5** The distributed representation of words and phrases [170].

In the cbow architecture, the current word  $w_i$  is predicted from its neighbouring words. Therefore all the neighbouring words are projected to the same position, which is similar to the BOW model in which the order of the words is ignored [171] (**Figure 2.5** (a)). In comparison, the skip-gram architecture predicts the surrounding words given the current word  $w_i$  (**Figure 2.5** (b)). It tries to adjust the synaptic weights in the NNLM so that the probability of classifying  $w_{i-n}, \dots, w_{i-1}$  and  $w_{i+1}, \dots, w_{i+n}$  as the observed neighbouring word given the current word  $w_i$  is maximised [171].

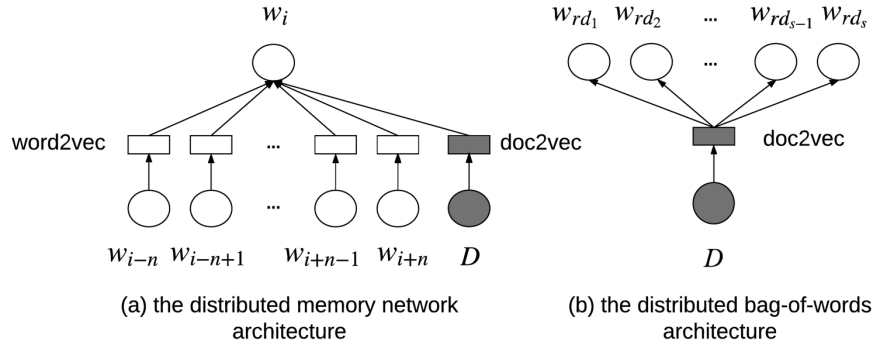
In the skip-gram architecture, as an example, for the word sequence  $w_1, \dots, w_i, \dots, w_T$ , the training object is to maximise the log probability defined as follows [170],

$$P = \frac{1}{T} \sum_{t=1}^T \sum_{-n \leq j \leq n, j \neq 0} \log p(w_{i+j} | w_t) \quad [\text{Eq. 2.3}]$$

where  $n$  denotes the number of upstream and downstream words that are predicted given the current word  $w_i$ . According to Mikolov, T., et al. (2013), more surrounding

words (larger values for  $n$ ) results in a better representation of the current words, which also incurs higher training time complexity [171]. Therefore, the hierarchical softmax [172] and the negative constructive sampling (NCE) [173] were introduced to make the training process feasible.

The distributed representation of words and phrases were later extended to sentences, paragraphs and documents [169]. By treating the document as a special word token and train it together with other words in the dictionary, the representation of the document is automatically learned from its composition words. This distributed representation of document also comes in two flavours, *i.e.* the distributed bag-of-words (dbow) and the distributed memory (DM) architecture. **Figure 2.6** demonstrates these two architectures for distributed representation of document  $D$  in comparison.



**Figure 2.6** The distributed representation of sentences and documents [169].

The DM architecture is similar to the CBOW architecture except that the document is regarded as a special word token and combined together with the each of the surrounding words  $w_{i-n}, w_{i-n+1}, \dots, w_{i+n-1}, w_{i+n}$  to predict the current words  $w_i$ . Therefore, the document representation can be regarded as the contextual information for each of its composition words. In the DBOW architecture, random words sampled from the document are predicted from the document representation. It is similar to the skip-gram architecture in which neighbouring words are predicted from the current word.

Compared to the generic word embedding techniques (including TF, TF-IDF and 1-of-K coding), the distributed representation has the following advantages. Firstly, it allows for condensed representation that are of fixed lengths. This is especially advantageous for the design of machine learning models which require fixed-lengths input features. Secondly, it is based on unsupervised learning and therefore does not require pre-labelled data, which greatly simplifies the step of data collection and dataset construction. Thirdly, the resulting representation is capable of encoding semantic meanings. According to the analogical reasoning task that was introduced by Mikolov, T., et al. (2013) [171], the question ‘Germany’: ‘Berlin’:: ‘France’:? got the answer ‘Paris’ given the distributed representations of the related words.

Similar to natural languages, biological sequences are typical sequential data and various natural language processing techniques have been applied to deal with biological sequences. For example, the 1-of-K encoding was used as the representation of protein sequences in different prediction tasks [78, 174, 175]. Therefore, it is straightforward to apply the distributed representation to protein sequences. Please refer to **Section 2.2.1** for existing distributed representations of biological sequences.

#### **2.1.4 Transfer learning, multitask learning and deep learning**

The lack of training data has been one of the main issues in machine learning methods. With more parameters involved in more complex structures, the deep learning models are more desperate for training data. It has become one of the bottlenecks for applying deep learning in many different areas.

The problem of lack of training data was partially leveraged by using two strategies which are related to each other. The first strategy is called *the multitask learning* where the parameters of the models for related tasks are shared with each other [176]. With different parameter sharing mechanisms, the task with less training data can be trained together with a related task that has sufficient training data and therefore compensates for the shortage of data. The other strategy is called the transfer learning, where the



knowledge gained in solving one problem is used to address another related problem [177]. These two strategies are similar in that they both employed the idea of parameter sharing.

In general, multitask frameworks in deep learning are implemented in two styles, one is called *hard parameter sharing* where multiple tasks share the same input layer and hidden layers and have their own task-specific fully connected layers and output layers [178, 179]. While the other is called *soft parameter sharing* where each task has its own weights for all hidden layers, and fully-connected layers. The distance between the parameters of these models are regularized to enforce similarity [180, 181]. Due to the diversity of the tasks that the multitask frameworks are applied to, there has been an interest in investigating the structural design of the multitask frameworks. More specifically, for different application scenarios, the number of shared layers, the number of task-specific layers and the way of sharing the parameters should be designed differently. To address this problem, the deep relationship networks were designed to explore the dependencies between corresponding task-specific layers in the hard parameter sharing framework [20]. The *cross-stitch multitask framework* was proposed to model the linear correlation between corresponding layers in the soft parameter sharing framework [182]. A joint many-task model for natural language processing was proposed to decrease the supervision by design a hierarchical architecture where the low-level tasks such as part-of-speech labelling and chunking were used as auxiliary tasks [183]. By applying these new techniques, the structure of the multitask framework was learned automatically depending on the correlation between the different tasks instead of manually specified beforehand.

Transfer learning was first introduced as the discriminability based transfer in 1933 [124]. It has been implemented using various machine learning methods including Bayesian networks [184], support vector machines [185] and decision forest [186]. With deep learning models, transfer learning is implemented in the following two ways. Firstly, deep learning practitioners release pre-trained models [187-190] based on which other

researchers can construct their own task-specific fully-connected layers or initialize their new model with the pre-trained weights. Secondly, feature extractors are designed and pre-trained which transform raw input data into condensed fixed-length representations. These representations are usually generated from unsupervised learning and are taken from the output of the last hidden layer of the designed model. In image processing, the AlexNet [191] produced a 4096-dimensional vector for each input image; and in natural language processing, the word2vec [170] and doc2vec [169] generate fixed-length vector representations for each input word and input document, respectively. With this pre-trained vector, even linear models can achieve satisfying performance. The distributed representation of words and documents is introduced in **Section 2.1.3**.

In this thesis, we applied the multitask framework and transfer learning for modelling and discovering the inter-correlation between different prediction tasks, which was in turn used for improving the prediction performance for individual tasks.

### **2.1.5 Open source platforms for constructing deep learning models**

The current commonly used deep learning platforms include the TensorFlow<sup>7</sup> by Google, Caffe<sup>8</sup> by UC Berkeley, Caffe2<sup>9</sup> by Facebook, Torch<sup>10</sup>, Theano<sup>11</sup>, Keras<sup>12</sup> and MXNet<sup>13</sup>. Most of the platforms are open sourced projects. To improve the training efficiency, these deep learning platforms usually support GPU and parallel computing. The most commonly used GPUs are provided by Nvidia which developed hardware (e.g. K40 and K80) and software (e.g. cudnn) specifically supporting high-performance deep learning computation. In this thesis, we developed all of the deep learning models using

---

<sup>7</sup> Tensorflow: <https://www.tensorflow.org>

<sup>8</sup> Caffe: <http://caffe.berkeleyvision.org>

<sup>9</sup> Caffe2: <https://caffe2.ai/docs/applications-of-deep-learning.html>

<sup>10</sup> Torch: <http://torch.ch>

<sup>11</sup> Theano: <http://deeplearning.net/software/theano/>

<sup>12</sup> Keras: <https://keras.io>

<sup>13</sup> MXNet: <https://mxnet.apache.org>

Tensorflow, with the hardware support from the Monash eResearch centre<sup>14</sup> which is configured with two GPUs in total.

## 2.2 Protein analysis based on deep learning techniques

In this thesis, we propose four deep learning frameworks for protein analysis, including a distributed representation of protein sequences, a deep learning framework for predicting protein structural and functional properties, two multitask deep learning frameworks for predicting two or more protein structural properties, and a deep transfer learning framework for predicting functional important sites. In this section, we introduce the existing representations of protein sequences and existing applications of deep learning techniques in protein analysis.

### 2.2.1 Existing representations of protein sequences

Protein sequences or amino acid sequences contain all the information for determining protein tertiary structures and functions. Therefore, protein sequences have been used directly and indirectly as the source of information for calculating various representation feature vectors. The direct way of using protein sequences is to generate amino acid-based representations including the amino acid indicator vectors [158], the 1-of-K coding [78], the sparse representations [192] and the distributed representations [193]. The indirect way of using protein sequences is to calculate or generate multiple categories of protein information including *evolutionary homology profiles* (such as position-specific scoring matrixes (PSSM) [8, 158, 194-196], HMMER profiles [158, 195, 196], HMM profiles), physicochemical properties (such as amino acid composition, polarity, hydrophobicity, surface tension, charge, normalized Van der Waals volume, and polarizability [69]), structural properties (protein secondary structures, solvent

---

<sup>14</sup> Monash eResearch centre: <https://www.monash.edu/researchinfrastructure/eresearch>

accessibilities, and torsion angles) and sequential patterns (such as curated sequential motifs [68]) from sequences of amino acids.

**Evolutionary homology profiles.** Functional regions or domains in proteins are conserved over time among different species whereas non-functional regions are free to changes [197]. Therefore, proteins that are evolutionarily homologous to each other may share related functions; and the structures and functions of a protein can be inferred from proteins that have similar evolutionary histories. Protein homology can be detected among two or more proteins from their sequences or structures. Due to the limited protein tertiary structure, sequence-based homology detection techniques were developed and widely applied in different protein analysis tasks.

Most of existing computational methods for detecting protein sequence homology are based on *sequence alignment*. With sequence alignment, biological sequences are compared so that similar regions are labelled out [99]. Sequence alignment can be categorised into local and global alignment and can be performed in a pairwise manner or among multiple sequences. Local sequence alignment seeks to label out all similar segments while global sequence alignments align two whole sequences. The previous is more applicable for dissimilar sequences that are suspect of similar segments, and the latter is often applied to generally similar sequences.

Sequence alignment has been widely used in a wide range of protein analysis tasks such as conserved domain identification (see **Section 1.2.3**) [198], phylogenetic analysis [199], protein homology profiling, and protein structural and functional predictions. In conserved domain identification, similar sequence segments among multiple proteins are labelled out as the conserved domain for a group of related proteins. In phylogenetic analysis, phylogenetic trees are generated from multiple sequence alignment, indicating the evolutionary distance between proteins and their common ancestors. In protein homology profiling, a scoring vector is generated for each amino acid at its specific position which encodes the pattern of occurring frequencies of amino acids. The

generated protein homology profile is widely used as one of the main features of protein structure determination and protein function classifications.

**Physicochemical properties.** Proteins are biological molecules with different chemical and physical properties, which, in combination, are referred to as physicochemical properties. The physicochemical properties of proteins are determined by the physicochemical properties of amino acids. The AAindex database recorded a set of 20 numerical values representing various physicochemical and biochemical properties of amino acids [200], while the PROFEAT tool provided online service for generating 10 sets of physicochemical and structural features from amino acid sequences, covering 1447 property values [201]. Note that the features here refer to *protein descriptors* that are calculated based on amino acid physicochemical property values, and van Westen, G.J. et al. (2013) [202] benchmarked 13 different protein descriptors that were calculated based on physicochemical properties, including z-scales, VHSE, and FASGAI.

Physicochemical properties are not always used in the form of protein descriptors. For some of the prediction tasks, specific physicochemical properties are selected to form feature vectors, depending on the prediction task itself. For example, in phosphorylation site prediction, average accumulative hydrophobicity has been demonstrated to be an important attribute of functionally important sites [17, 203]. Therefore, it is calculated independently from other physicochemical properties and used as one of the sources of information for generating the input feature.

**Structural information.** In protein structural and functional prediction, protein structural and functional properties are inter-correlated and mutually evidential. Therefore, it is not uncommon to use predicted protein structural information as the input feature for predicting related protein structures or functions. Some of the most commonly used protein structural information include protein secondary structures, protein surface accessibilities, and protein intrinsic disorder. Ideally, accurate structural information can be derived from determined tertiary structures. However, in most cases, structural information is predicted directly from protein sequences using existing state-of-the-art

bioinformatics tools. For example, homology profiles and physicochemical properties are widely used as input features for protein structure prediction [101, 194], while predicted protein secondary structures have been used as part of the input feature for protein intrinsic disorder prediction [158]. For detailed introduction of bioinformatics tools, please refer to **Section 1.2.5**.

In the thesis, we refer to the above three sources of representations, including homology profiles, physicochemical properties, and the structural information, which are generated from protein sequences with biological heuristics, as the biological representation of protein sequences. They are widely researched and used in protein sequence-based predictions due to their strong biological backgrounds.

**Sequence motifs and protein sequence-based representations.** A protein sequence motif, or pattern, can be broadly defined as a set of conserved amino acid residues that are important for protein function and are located within a certain distance from each other [204]. The PROSITE database [68] hosted the most extensive and most comprehensive collection of protein sequence motifs that can be used for various protein analysis purposes. As for protein sequential representations, 1-of-K coding [78] and the sparse representation [192, 205] have been proposed to indicate the amino acid type at each position in a protein sequence.

**The continuous distributed representation of biological sequences.** Based on the distributed representation of words and phrases, Asgari, E. and Mofrad, M.R.K. (2015) proposed a continuous distributed representation of biological sequences [193], named as *ProtVec*. The vector representation of each protein (or biological) sequence was generated by summing up the distributed representation of biological words.

Let  $S$  denote the protein sequence whose representation is queried. The distributed representation of  $S$  is generated in the following three steps. Firstly, the protein sequence  $S$  is split into a list of  $n$ -grams, denoted as  $S = w_1, w_2, \dots, w_n$ . To demonstrate an analogous mapping between protein sequence and natural language sentences, in this thesis, we refer to each of the  $n$ -gram as a *biological word*. Secondly, the distributed

representation of each of the biological word was generated from a pre-trained generative model trained based on the word2vec algorithm (see **Section 2.1.3**). Thirdly, the distributed representations of the words  $w_1, w_2, \dots, w_n$  are summed up to form the distributed representation of the protein sequence  $S$ .

The idea of this generation strategy is similar to that of the Bag-of-Word model (see **Section 2.1.3**) where the representations of protein sequences are generated ignoring of the order of its biological words. In **Chapter 4 – Chapter 8**, we performed both qualitative and quantitative comparisons between ProtVec and the distributed representation implementations proposed in this thesis, demonstrating the advantages and disadvantages of these two implementations in different application scenarios.

### 2.2.2 Limitations of the application of existing protein representations

Existing biological representations of protein sequences have been applied to various protein sequence-based predictions<sup>15</sup>. They are conducted in three different granularities, including the whole sequence-level predictions, the residue-level predictions and the biological interaction-level predictions. Here, whole protein sequences represent more abstractive level than individual residues, while biological interactions usually involve two or more protein sequences, which represent a higher level of abstraction. Due to the construction principles of existing biological representations, they are designed to represent protein sequences in only one of the three granularities. The following cases demonstrate the potential issues in applying biological representations to protein sequence-based predictions.

- The homology profile position-specific scoring matrix (PSSM), for example, is constructed at the residue-level where each residue is represented as a vector

---

<sup>15</sup>. In this thesis, protein sequence-based predictions refer to the computational estimation, identification, or determination of protein structural and functional properties that are expensive to determine with experimental methods.

of size 21. However, to represent whole protein sequences, the vectors of all residues have to be either 1) summed up to form a one-dimensional vector of size 21 or 2) concatenated to form a 2-dimensional matrix representation where the size of the first dimension corresponds to the length of the protein sequences. With solution 1), the resulting representation ignored the order of amino acids of the protein sequence, and it is biologically meaningless to sum up the evolutionary scores of all residues. With solution 2), however, the lengths of protein sequences may vary from 3 to 35,000, which results in protein sequence representations with dramatically different sizes.

- The physicochemical property protein solubility, as an example, is defined at the whole-sequence level. The surface of protein has a net charge that depends on the number of identical charged amino acids on the protein surface, and the pH [206]. This net charge is considered to be closely related to the solubility of the protein. Therefore, the estimation of protein solubility is conducted at a level more abstractive than individual residues. Its calculation has to be at the level of whole-sequence for both proteins and peptides. Physicochemical properties like protein solubility are not applicable for residue level predictions and biological interaction level predictions.
- Finally, the existing distributed representation of protein sequences, ProtVec, was proposed to map each protein sequence to a vector representation. Therefore, it can only be applied to whole sequence-level prediction tasks such as protein family prediction [193].

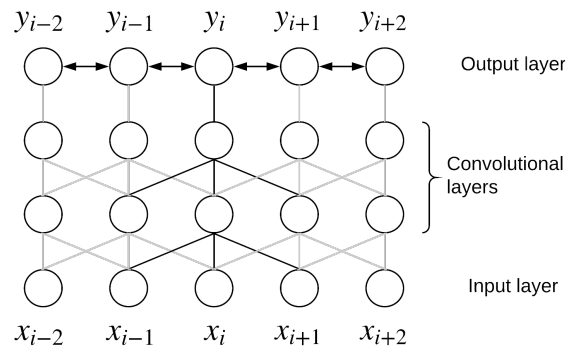
### **2.2.3 Existing applications of deep learning to protein analysis**

Existing application of deep learning has achieved state-of-the-art performance for various prediction tasks in protein sequence-based prediction tasks, such as protein secondary structure, solvent accessible area and torsion angle prediction, protein



intrinsic disorder prediction, phosphorylation site prediction, residue contact map prediction, protein function prediction, and protein-protein interaction prediction.

Deep convolutional neural fields (DeepCNF) represents one of the most widely applied deep learning techniques in protein sequence-based predictions. For protein secondary structure prediction, Wang, S. et al. (2016) proposed the DeepCNF-SS model to capture local correlation between input features and output labels [156]. The input features were constructed as the concatenation of the 1-of-K encoding for 21 amino acids and the 21-dimensional homology profile (PSSM) generated using PSI-BLAST [99]. The resulting DeepCNF-SS model achieved the state-of-the-art performance for protein secondary structure prediction with an accuracy score of ~84% for the 3-class DSSP categorical system and ~72% for the 8-class DSSP categorical system (see **Section 1.2.3**). In **Figure 2.7**, the DeepCNF-SS model is composed of an input layer for 5 residues centred at the target residue, two convolutional layers with filter size 3, and an output layer modelled as the conditional random fields for the output labels of the 5 input residues.



**Figure 2.7** The DeepCNF-SS model for protein secondary structure prediction [156].

Based on the same DeepCNF model, Wang, S. et al. proposed the DeepCNF-D model [158] and the AUCPreD model [207] for predicting protein intrinsic disorder. The former assigned a reciprocal weighted ratio between ordered and disordered examples to address the issue of imbalanced dataset [158], while the latter used a refined objective

function to maximise the AUC score [207]. Please refer to **Table 2.1** for detailed description of their feature groups.

Different from deep CNF, deep CNN only models the local correlation within the input feature. Yet, it has achieved the state-of-the-art performance in several prediction tasks such as phosphorylation site prediction. For phosphorylation site prediction, the MusiteDeep employed a combination of convolutional network and the attention-mechanism to automatically search for important positions in the input sequence [78]. The input feature was constructed as the 1-of-K encoding of the raw input sequences, which ignores the abundant information that are encoded in homology profiles, physicochemical properties and structural information, but has the advantage of significantly shorter time for sequence pre-processing. Interestingly, the idea of transfer learning was mentioned in MusiteDeep to reuse the pre-trained models of S, T, Y sites as the feature extractor of predictive models for specific kinases. It was inspired by the fact that there are more sufficient training data in general phosphorylation site prediction than in kinase-specific phosphorylation site prediction. However, there is not incorporation of the hierarchical classification system of protein kinases.

Another deep learning technique that has been applied to biological data is the RNN [118], especially with the LSTM implementation [117]. For IDP/IDR prediction, Hanson, J. et al. (2017) proposed to use a three hidden-layer bidirectional recurrent neural network to predict the disordered states of the target residue, based on the feature extracted from its neighbouring residues [153]. In this architecture, the first hidden layer was implemented as a fully-connected layer, while the latter two were implemented as bidirectional LSTM units. The input features were constructed from evolutionary profiles, physicochemical properties and predicted structural information.

Other deep learning architectures including fully-connected deep neural networks, deep restricted Boltzmann machines (RBM), and deep belief nets (DBN) are used in protein analysis as well. For protein torsion angle prediction, Li, H. et al. (2017) systematically evaluated the prediction performance of four different deep learning

models, including deep CNN, deep RNN, deep DBN and deep RBM [208]. For contact map prediction, Di Lena, P., Nagata, K. and Baldi, P. (2012) proposed a three-step procedure to predict coarse contact and orientation, element alignment, and residue-residue contact in a sequential order [64]. The first step briefly classifies pairs of secondary structures as parallel contact, anti-parallel contact, and no-contact using 2-dimensional RNN. The second step assigns contact probabilities to pairs of residues in contacting secondary structures. And the third step predict accurate residue-residue contacts based on the previous two steps using a 2-dimentional feed-forward neural network.

#### **2.2.4 Limitations of existing applications of deep learning to protein analysis**

These applications of deep learning in protein analysis are mostly conducted in an *ad hoc* manner. With improving the prediction performance as the ultimate goal, there is a lack of systematic analysis of the performance of deep learning in protein analysis, investigating why deep learning models performed superior or inferior to other methods, the heuristics from the biology that can be used for the designing of deep learning models, and the relation between different prediction tasks from the perspective of predictive model design.

### **2.3 Existing methods for protein prediction tasks**

To demonstrate the usefulness of our proposed deep learning frameworks, we applied the proposed deep learning models to five prediction tasks, including protein family prediction, DTI prediction, phosphorylation site prediction, protein SSP prediction and IDP/IDR prediction. In this section, we introduce the existing methods for each of the above five prediction tasks and summarise the advantage and disadvantage of these existing methods, providing comprehensive research backgrounds and baselines to which we compare the frameworks proposed in the following chapters.

### 2.3.1 Protein family prediction

According to the convention, protein families refer to groups of proteins with the common origin of evolution, reflected by their related functions or similar sequences and structures. For proteins with little experimental evidence, the identification of their protein family can provide additional information of protein structures and functions. This identification relies on the construction of the protein family database. Currently, these protein family databases are constructed via the combination of the expert curation and computational prediction. These protein family databases include the Pfam database [70], the PROSITE database [68], the InterPro database [86], and the PIRSF database [84]. Among these four databases, the InterPro database and the PIRSF database are meta-databases that combine annotation from multiple databases for more complete collections of protein family annotations, while Pfam and PROSITE proposed specialised methods to identify family members of different protein families.

**HMMER.** The Pfam database is constructed using the HMMER tool [87], which generates a *Hidden Markov Model (HMM) profile*, for each protein family, from the multiple sequence alignment of existing protein family members. In 1998, Eddy, S. R. (1998) proposed the profiling biological sequence using HMMs. A typical HMM is composed of a sequence of hidden states each of which has an emission probability for generating the next element in sequence and a transition probability determining whether transit to the next state. The training process of such a model is to learn the emission and transition probabilities of each state given the observed sequences of elements that are generated from the model. Once the emission and transition probabilities are learned, the model can be used a) to generate new sequences; and b) to check whether a new sequence would be generated from the model.

Protein family prediction is benefitted from the second application scenario where a new protein sequence is checked against a pre-trained protein family HMM profile to determine whether it is a member of this protein family. In practical, the emission and

transition probabilities of a protein family HMM profile can be learned from the protein sequences of existing protein family members by using supervised learning algorithms such as gradient descent, Gibbs sampling, simulated annealing and genetic algorithms. They can also be calculated directly from the statistics in multiple sequence alignment of the protein family members.

HMMER represents the early attempts to model protein sequences using sequential models. Once the model is trained, HMMER can recognise a member sequence of a specific protein family within linear time. It has been used to generate homology-based features for protein sequences in multiple prediction tasks.

**PROSITE.** The PROSITE database was constructed based on motif descriptors that were either represented as patterns or profiles. In PROSITE, the patterns of a protein family refer to the regular expression indicating the residue pattern for evolutionarily conserved domains. These conserved domains are usually very important signatures for recognising the members of a protein family, including enzyme catalytic sites, prosthetic group attachment sites, cysteines involved in disulphide bonds, and regions involved in binding activities [68]. All patterns were manually constructed and curated.

The other motif descriptor, the profiles, was constructed based on the sequence profile introduced by Gribskov, M., et al. (1987) [209]. This sequential profile is proved to be equivalent to the HMM-based profile [210], and it functions in the same way that the HMM profile does. By combining the pattern-based descriptor and the profile-based descriptor, PROSITE was able to recognise both the single short domains that are conserved with certain amino acid patterns and the general domains in proteins sequences that are lack of amino acid patterns but demonstrate similar profiles in their contextual sequences or structures.

In this thesis, we analysed a new protein representation generated based on the distributed representation of protein sequences. We constructed protein family classifiers based on this new protein representation and analysed the advantages and

disadvantages compared to the existing protein patterns and profiles introduced in this section.

### 2.3.2 Intrinsically disordered protein prediction

During the process of protein expression, not all proteins (and regions) form stable three-dimensional structure during its lifetime. Proteins and protein regions that do not form stable structures are referred to as intrinsically disordered proteins (IDPs) and regions (IDRs). IDP/IDRs have been proven to be related to the mechanism of multiple diseases. Therefore, it is crucial to identify IDP/IDRs. Multiple computational methods have been identified to predict IDP/IDRs automatically from protein sequences.

In this thesis, we propose to use a multitask deep learning framework to predict IDP/IDRs together with protein SSPs, and conducted comprehensive experiments to compare the prediction performance of the proposed deep learning framework to existing methods including PrDos2 [211], DisoPred3C [102], MultiCom, SPINED-D [212], PrDos-CNF, DisoPred3 [101], BioMine and DeepCNF-D [158].

**PrDos2.** The ProDos2 method combines the disordered protein/region prediction results of two predictors, the sequence-based predictor and the template-based predictor. The sequence-based predictor takes the calculated sequence profile PSSM as the input features, based on which SVM models are trained to predict whether a target residue is disordered or not. The template-based predictor assigns disorder tendency to each target residue according to the intrinsic disorder of aligned proteins whose structures have been determined. The alignment of the target protein  $P$  against the RCSB PDB database is calculated using PSI-BLAST, and the disorder tendency for target residue  $r_i$  in  $P$  is calculated as follows [211],

$$P_i = \frac{\sum_{j=1}^n \partial_j I_j}{n} \quad [\text{Eq. 2.4}]$$

where  $n$  represents the number of aligned proteins,  $I_j$  indicates whether the  $j$ -th protein is aligned with the current protein  $P$ , and  $\partial_j$  is set to 1 if the aligned residue in the  $j$ -th protein is disordered.

**DisoPred3C.** The DisoPred3C method [102] predicts protein intrinsic disordered based on the sequence profiles that is calculated from the PSI-BLAST tool. It is similar to the sequence-based predictor in PrDos2, except that PrDos2 used a sliding window of size 27 centred at the target residue, while DisoPred3C used a sliding window of size 7.

**SPINE-D.** The SPINE-D method [212] employed a multilayer neural network with two hidden layers of 51 neurons units and a filter layer of 11 neurons units. It produces three outputs for ordered residues, disordered residues in short regions and disordered residues in long regions. The latter two were summed up as the output for disordered residues. As for input features, SPINE-D combined features from sequence profiles, predicted structural information, physical parameters, and N- and C- terminus indications.

**DisoPred3.** The third version of DisoPred combined the outputs of three prediction components, an SVM classifier, a neural network classifier, a nearest neighbour classifier, to specifically improve the prediction performance for long disordered regions. Different from other methods, the DisoPred3 method also provided the prediction of protein-binding sites for disordered regions.

**DeepCNF-D.** The DeepCNF-D methods employed the aforementioned deep convolutional neural fields which extract high abstract representation using convolutional hidden layers and models the local correlation among predicted neighbouring disordered states. The input features for the DeepCNF-D method is constructed from sequence profiles, predicted structural information, and physicochemical properties.

**AUCpreD.** The AUCpreD methods also employed the deep convolutional neural fields that are used in DeepCNF-D but with a different objective function that is used to

train a model for maximising the AUC score. A polynomial approximation of the AUC score was used, whose gradient is calculated and used as the objective function.

**SPOT-disorder.** The SPOT-disorder disorder represents the more recent state-of-the-art method for disordered protein/region prediction. It employed the long-short memory network to predict long disordered regions. It also used input features computed from sequence profiles, predicted structural information and physicochemical properties. Similar to SPINE-D, the SPOT-disorder method also separated the prediction of short disordered regions from long disordered regions.



**Table 2.1** A summary of machine learning methods for IDP/IDR prediction.

Predictors	Algorithms	Data sources	Feature groups
PrDos2	SVM	-	Sequence profile, and templates-based alignment
DisoPred3C	SVM	PDB [40]	Sequence profile
DisoPred3	SVM, neural network, KNN	DisProt v5.0 [89], RCSB PDB [40]	Outputs from the SVM, the neural network, the KNN classifier, N- and C-terminus indications
SPINE-D	Two-layer neural network	DM4229 [212]	Sequence profiles, structural information, physical parameters, N- and C-terminus indications
DeepCNF-D	DeepCNF	Disoder723 [213]	Sequence profiles, predicted structural information, and physicochemical properties
AUCpred	DeepCNF	UniProt90 [92]	Sequence profiles, predicted structural information, and physicochemical properties
SPOT-disorder	LSTM [117]	DM4229 [212]	Sequence profiles, the Shannon entropy, predicted structural information, and physicochemical properties

### 2.3.3 Phosphorylation site prediction

Phosphorylation is one of the most important protein post-translational modification. Multiple computational methods have been proposed to predict phosphorylation sites in order to save *in vivo* or *in vitro* experimental resources. In this thesis, we propose the context-based prediction of phosphorylation site using the distributed representation of protein sequences. To demonstrate and validate the performance of the proposed method, we conducted extensive comparison experiments among different phosphorylation site predictors. In this section, we introduce some of the state-of-the-art predictors for phosphorylation sites, including the group-based prediction system (GPS), Musite, KinasePhos, NetPhos, PhosphoPick and PhosphoSVM.

**GPS.** The GPS method [77, 214, 215] proposed to classify potential sites into a hierarchy of kinase tree structure. It clusters these sites according to the homology similarity among their neighbouring residues, which was defined as the phosphorylation site peptide  $PSP(m, n)$  where  $m$  and  $n$  refer to the number of the upstream residues and downstream residues respectively [215]. The homology similarity was assumed to represent the 3D structural similarity and therefore functional similarity between two  $PSP$ . The amino acid substitution matrix BLOSSUM62 was used to calculate this homology similarity.

**MusiteDeep.** The MusiteDeep method [78] is the only existing method that is constructed based on deep learning architecture. It employed a three-layer convolutional neural network, based on which a two-dimensional attention mechanism and a fully-connected layer were used to map the output of the last hidden layer to the final outputs. This method takes the raw protein sequence as the direct input, which is encoded using the 1-of-K coding. Therefore, no feature calculation was needed. This method represents the state-of-the-art performance for phosphorylation site prediction at the moment.

**Musite.** The Musite method [216] proposed to predict phosphorylation sites based on three sets of features including the k-nearest score, protein disordered states and

amino acid frequencies among which K-nearest scores describe the local sequential patterns, protein disordered states were considered to improve the discrimination between phosphorylation and non-phosphorylation sites [217], and the amino acid frequencies encodes the residue-level information. The support vector machine (SVM) was used as the model for prediction. It addressed both general and kinase-specific phosphorylation site prediction, providing prediction models for more than 13 kinases.

**KinasePhos.** The KinasePhos method [218, 219] proposed to predict kinase-specific phosphorylation sites using the HMM profile and the coupling pattern of the surrounding sequence segment, based on the SVM classifier. The HMM profile encoded the homology similarity between local sequence segments while the coupling pattern was a new feature proposed to describe the amino acid pattern around the potential phosphorylation sites.

**NetPhos.** The NetPhos method [35, 154] was one of the earliest predictors proposed to computationally predict post-translational modifications. It used a neural network to combine sequential and structural motifs in a unified prediction model.

**PhosphoPick.** The PhosphoPick method [220] was the only one predictor that incorporate the protein interaction networks for computing the similarity between different local sequence segments of potential phosphorylation sites. It can be combined with other methods to further improve the prediction performance.

**PhosphoSVM.** The PhosphoSVM methods [221] was recently proposed to combine multiple features in a unified framework to better predict general phosphorylation sites. The features can be categorised into five categories, *i.e.* homology-based features such as the Shannon entropy, the relative entropy and the K-nearest scores, physicochemical property-based features such as the Taylor's overlapping properties and accumulative hydrophobicity scores, protein secondary structures, surface accessibility area and protein disordered states. All the seven features were concatenated for training corresponding SVM models for S, T and Y sites respectively. PhosphoSVM only predicts general phosphorylation sites.

**Table 2.2** A summary of machine learning methods for phosphorylation site prediction.

Predictors	Algorithms	Data sources	Feature groups	G/K*
GPS 3.0	Hierarchical clustering [214]	Phospho.ELM [94], PhosphoBase [222]	Phosphorylation site peptide (PSP) sequence similarities [215]	G/K
MusiteDeep	Deep CNN	UniProt/Swiss-Prot, RegPhos [223]	Raw amino acid sequences	G/K
Musite 1.0	Ensemble learning [224]	Phospho.ELM, UniProt, PhosphoPep [225], PhosphAt [98]	K-nearest neighbour (KNN) scores [216], disorder states and amino acid frequencies [217]	G/K
NetPhos 3.1	Neural networks	PhosphoBase	Convolutional sparse coding [155] of local sequence contexts	G/K
KinasePhos 2.0	SVM [145]	PhosphoBase, Swiss-Prot/UniProt [39]	Local sequence patterns and local coupling patterns [219]	K
PhosphoPredict	RandomForest	Phospho.ELM	Sequence, structural and functional features	K
PhosphoPick	Bayesian networks [226]	Phospho.ELM, HPRD [227]	Protein-protein interactions [228] and protein cell-cycle types [229]	K

\* G/K denote general and kinase-specific phosphorylation site prediction respectively.

#### 2.3.4 Drug-target interaction prediction

Discovering drug-target interactions has important implications for drug development, drug repositioning [81] and side effect detection [230]. Despite their accurate results, experimental investigations using *in vitro* and *in vivo* techniques for predicting DTI are usually time-consuming and labour intensive. As an alternative, *in silico* methods have been applied to identify potential interactions in a high-throughput and cost-effective manner [231]. According to the methodology applied, existing *in silico* DTI prediction methods can be categorised into 3 types: docking simulation-based [232], machine learning-based [233-235] and network analysis-based [81] methods. Docking simulation is a technique where drugs dock into specific positions of protein 3D structure. This method requires the calculation of docking energy based on the tertiary structures of the protein targets. Therefore it is only applicable for proteins whose tertiary structures have been identified [231]. Network analysis is a technique that investigates relations between nodes based on graph theories and network analysis, which is only applicable to drugs and proteins that have known interactions. It is more difficult to predict novel DTIs by using network analysis-based methods. In this section, we mainly focus on machine learning-based methods, which provide novel predictions and insights of potential DTIs.

Due to the network structure of drug-target interactions, the similarity assumption was widely adopted among computation methods for DTI prediction. It assumes that similar proteins are more likely to interact with the same drug and vice versa. Therefore, features for DTI predictions are usually constructed as similarity adjacent matrixes indicating the similarity between two proteins or two drugs in terms of some pre-defined distance metrics. We categorise these protein features and drug features into four types with respect to the source of information for constructing them.

**Sequence or chemical structure-based features.** Sequence or chemical structure-based features are the most widely used features [233, 235-240], which encode the information from basic properties of proteins or drugs, e.g. amino acid sequences for proteins and chemical structures for drugs. For protein sequences, the Smith-Waterman score [241] was used to calculate the similar regions between two amino acid sequences. While for drug chemical structures, the SIMCOMP tool [242] was used to calculate the maximum common sub-graph, where a drug's graph is constructed with atoms as nodes and covalent bonds as edges. Information required for constructing sequence or chemical-based features is usually easy to obtain directly from aforementioned data sources like DrugBank [97], KEGG [96], Swiss-Prot/UniProt [39], RCSB PDB [40], etc. For drug chemical structure information, drug descriptors can be generated as a canonical representation of the structure, which is later used for feature calculating. Different molecule fingerprints include Daylight [243], ECPF [244], Spectrophores [245], etc.

**Protein inter-domain -based features.** Protein inter-domain -based features refer to information extracted from relevant databases for proteins or drugs. Instead of using basic information like amino acid sequence or chemical structure, proteins' or drugs' interactions with other domains are used as information for calculating features. For example, drugs sharing similar correlated information from other domains, such as side effect, receptor families, gene expression responses are more likely to target similar proteins; while proteins that are closer in a protein-protein-interaction (PPI) network are more likely to be targeted by similar drugs [246]. Here information such as side effect, receptor families, gene expression and PPI are from other domains that are related to drugs or proteins. Multiple data resources, such as Sider [247] for side effect and BioGrid [228] for PPI, should be integrated and cross-referenced to compute the features.

**Annotation-based features.** Annotation-based features refer to features calculated based on different classification systems. For drugs, Anatomical Therapeutic Chemical (ATC) [248] annotations and the KEGG BRITE ontology [96] are widely used as

classification schemes; while for proteins, Gene Ontology (GO) [249] and Enzyme Commission number (E.C. number) [250]. These classification systems are usually organised as hierarchical structures where drugs or proteins are attached to one or more tree nodes. According to Resnik's semantic similarity algorithm [251, 252], it is assumed that two drugs or two proteins that share more common ancestors are more similar to each other. The algorithm assigns weights to classification tree nodes according to their population in data sources, such as KEGG and UniProt, and calculates the drug or protein similarity accordingly.

**Interaction-based feature.** The interaction-based features are constructed from partially known drug-target interactions, which is referred to as interaction profile by van Laarhoven, T., et al. (2011) [236]. It is suitable for situations where some of the interactions are known, and more interactions can be predicted for existing proteins and drugs. The partially available DTIs are represented as binary feature vectors for each drug and protein, e.g. 10101... and the basic idea is that drugs (proteins) sharing similar interaction profiles are more likely to target similar proteins (drugs) [236, 253]. This information can also be combined with other features of drugs or proteins to predict DTIs between new drugs and proteins [237].

**Table 2.3** summarises the four groups of features that are used in existing methods for DTI prediction.

**Table 2.3** Four categories of features for drug-target interaction prediction.

Name	Description	Domain
<i>Sequence or chemical structure-based</i>		
Sequence	Similarities between protein sequences are calculated with Smith-Waterman scores.	Protein
Mismatch	Similarities between protein sequences are calculated as the common short sequence of amino acids up to some mismatches.	Protein
Chemical	Size of common substructures between two compounds calculated with SIMCOMP.	Drug
<i>Protein inter-domain -based</i>		
PPI	Calculated based on proteins' distance within a PPI network.	Protein-Protein
Ligand	Similarities between drugs' receptor families are calculated using the Jaccard score.	Drug-Ligand
Gene Expression	Similarities between the genes that respond to drugs calculated using Spearman's rho, Jaccard score and Gene Set Enrichment Analysis (GSEA).	Drug-Gene
Side effect	Similarities between drugs' side effect set are calculated using the Jaccard score.	Drug-Side effect
<i>Annotation-based</i>		
Hierarchical	Number of common ancestors in EC tree and classification tree.	Protein
GO	Similarities between proteins' annotation in Gene Ontology hierarchical tree are calculated with the semantic similarity score.	Protein
ATC	Similarities between drugs' ATC code in ATC hierarchical tree are calculated with the semantic similarity score.	Drug
<i>Interaction-based</i>		
Interaction profile	No similarity matrix is generated. The known interactions between protein and drugs are encoded as binary vectors.	Protein-Drug



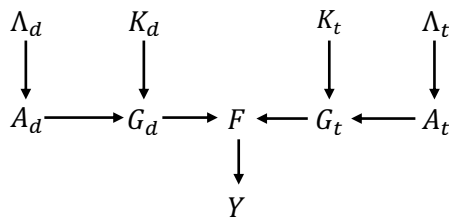
Various machine learning approaches have been applied to DTI prediction, including kernel-based methods [240] [236] [254] [236], the local model [236], the K-nearest neighbor [237], Bayesian networks [233], statistical relational models [234], the clustering method [255] and neural networks [253]. In this section, we introduce 7 representative prediction methods which respectively employed the above machine learning approaches for the overall analysis of existing DTI prediction. They are the pairwise kernel models (PKM) [254], bipartite local models (BLM) [236], Weighted Nearest Neighbor profile (WNN) [236], Bayesian matrix factorization (KBMF2K) [236], Probabilistic soft logic (PSL) [234], Semantic Edge Partitioning (SemEP) [234] and Restricted Boltzmann machine (RBM) [253].

**Pairwise kernel model.** The pairwise kernel model (PKM) [254] is a machine learning model that incorporates information from both drug and protein space as pairwise kernels for DTI prediction. It formalised a pairwise kernel which encodes the combined information of a pair of a drug and a protein using tensor calculation. In most cases, the drug kernel  $K_{drug}(d, d')$  or protein kernel  $K_{target}(t, t')$  was calculated as the similarity between two drugs or two proteins [233, 239, 240].

**Bipartite local model.** The bipartite local model (BLM) [235] is an extension of the bipartite graph representation by using local models. It predicts targeting drugs for a given protein and target proteins for a given drug, by training a binary classification local model for each protein and drug respectively. Therefore, BLM is relatively time-consuming. In addition, it is not applicable to predict the interaction between a drug and a protein if the interaction profiles of neither the protein or the drug is available

**Bayesian matrix factorisation.** Bayesian matrix factorisation (KBMF2K) [233] is a Bayesian network-based method. The constructed Bayesian network in **Figure 2.8** performs three processing steps including the dimensionality reduction, the matrix factorization and the binary classification. The step of dimensionality reduction step projects the drug representation  $A_d$  and the protein representation  $A_t$  to their

corresponding representations  $G_d$  and  $G_t$  in the same feature space, using the drug kernel function  $K_d$  and protein kernel function  $K_t$  respectively. The step of matrix factorization produces the inner product  $F$  of the mapped representations. And, finally, the binary classification step outputs the predicted label  $Y$  indicating whether there is interaction between the input drug and protein.



**Figure 2.8** Bayesian matrix factorization for drug-target interaction prediction [233].

**Gaussian Interaction Profile.** Gaussian Interaction Profile (GIP) refers to the feature vector calculated from the interaction profiles of an existing drug-target interaction network. GIP-based predictor [236] explores new interactions between drugs and proteins in this existing interaction network. Therefore, it only predicts DTIs for drugs (or proteins) with known DTIs.

**Gaussian Interaction Profile-weighted nearest neighbour.** The limitation of GIP had been overcome by the weighted nearest neighbour (WNN) [237], which can predict DTIs for drugs (or proteins) with no interaction information. The rationale is that a new drug's or a new protein's interaction profile can be calculated as the weighted sum of its similar drugs or proteins' interaction profiles.

Let  $y_1 \dots y_{n_d}$  be the interaction scores between target  $t$  and drug  $d$ 's top  $n_d$  chemically similar drugs, in the decreasing order of their similarity with  $d$ . The interaction score of the drug  $d$  is defined as follows,

$$y_{WNN}^d = \sum_{i=1}^{n_d} \omega_i y_i \quad [\text{Eq. 2.1}]$$

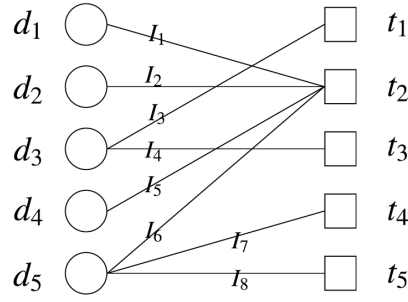
where  $\omega_i = T^{(i-1)}$ ,  $T \leq 1$  denotes a decay value used to modulate the weight of each similar drugs.

GIP-WNN is a highly effective and efficient method for predicting novel DTIs between drug  $d$  and protein  $t$  as long as the interactions of their similar peers are available.

**Probabilistic soft logic.** Probabilistic soft logic (PSL) [234] is a statistical relation model that combines logic and probabilistic graphical model. Let  $D_i$  and  $T_j$  represents the  $i$ -th drug and  $j$ -th protein, respectively, and *SimilarTarget*, *SimilarDrug*, *Interacts* represents protein similarity, drug similarity and drug-target interaction, respectively. With similarity measurements  $\alpha$  and  $\beta$  specified, the PSL models drug-target interaction domain knowledge [256] which can be described using following weighted first-order logic formulae, where weights  $\omega_1$ ,  $\omega_2$  and  $\omega_3$  describe the correctness of each of the rules.

$$\begin{aligned}\omega_1 \quad & \text{SimilarTarget}_\beta (T_1, T_2) \wedge \text{Interacts}(D, T_2) \rightarrow \text{Interacts}(D, T_1) \\ \omega_2 \quad & \text{SimilarDrug}_\alpha (D_1, D_2) \wedge \text{Interacts}(D_2, T) \rightarrow \text{Interacts}(D_1, T) \\ \omega_3 \quad & \text{SimilarDrug}_\alpha (D_1, D_2) \wedge \text{SimilarTarget}_\beta (T_1, T_2) \wedge \text{Interacts}(D_2, T_2) \rightarrow \text{Interacts}(D_1, T_1)\end{aligned}$$

**Semantic-based edge partitioning.** Semantic-based edge partitioning (SemEP) [255] is a clustering process that is solved as an optimization problem with two targets, *i.e.* maximizing the similarity density within each cluster and minimizing the number of. When applied to DTI prediction, the similarity density consists of 3 parts: a) the average score of interaction edges; b) the average drug-drug similarity scores; and c) the average protein-protein similarity scores.



**Figure 2.9** An example of applying semantic-based edge partitioning to drug-target interaction prediction.

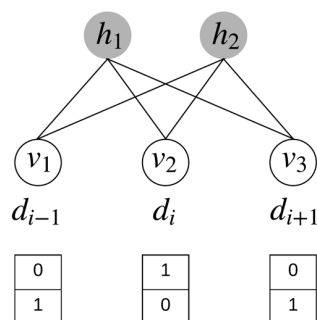
**Figure 2.9** demonstrates an example of drug-target interaction prediction using SemEP, where drug  $d_1 - d_5$  and protein  $t_1 - t_5$  have partially known interactions  $I_1 - I_7$ . The corresponding similarity densities are calculated as follows,

$$sDensity_1 = 1 + avg(s(d_1, d_2) + s(d_2, d_3) + s(d_1, d_3)) + avg(s(t_1, t_2) + s(t_2, t_3) + s(t_1, t_3))$$

$$sDensity_2 = 1 + avg(s(t_2, t_4) + s(t_2, t_5) + s(t_4, t_5))$$

By maximizing  $sDensity_1 + sDensity_2$  and minimising the number of clusters (which is 2) a grouping of drugs and proteins, *i.e.*  $(d_1, d_2, d_3, t_1, t_2, t_3)$  and  $(d_4, t_2, t_5)$ , are formed. New predictions are formed between each pair of drug and target within each cluster.

**Restricted Boltzmann Machine.** Restricted Boltzmann Machine (RBM) [253] was the first approach proposed to predict drug-target interaction with different modes of action, *i.e.* binding, activation and inhibition. The predictive model is a two-layer (one visible layer and one hidden layer) undirected graphical model, where each visible unit is connected with each hidden unit while there is no intra-layer connection. The hidden layer is used to model the latent features supporting interaction prediction, and the visible layer represents observations of known interactions. **Figure 2.10** provides a simple example of how the model is working.



**Figure 2.10** RBM template for predicting interactions of individual protein  $t$  [253].

The advantage of these graphical models is that joint inference [127] is performed to collect evidence from different features. However, it is non-trivial to decide how many hidden units are appropriate and to interpret these hidden units.

**Table 2.4** lists a summary of machine learning-based methods for drug target interaction prediction. Most previous works only choose one pair of drug/protein features (usually the best pair of similarity matrixes at the time) as an example to demonstrate the experiment performance. However, from the perspective of machine learning models, KRM, BLM, KBMF2K, PKM, GIP, GIP+WNN, SemEP and RBM can take any pair of drug/protein feature while PSL can take multiple pairs of features simultaneously as the input feature.

**Table 2.4** A summary of machine learning based methods for DTI prediction.

Methods	Features	Models	Pros and cons
PKM	Mismatch, local alignment, hierarchical	Support vector machine	It is designed to take any kernels for drug and proteins. It is a useful framework for comparing different features.
BLM	Substructure; sequence	Local model	It transforms edge prediction problem to simple classification problem, but it cannot deal with drugs or proteins without available interaction information.
KBMF2K	Substructure; sequence	Bayesian theory	Priors are set as the hyper-parameters to control the projection from kernel matrix to inner product instance.
SemEP	Substructure; sequence	Clustering	It is the only unsupervised learning method and time-efficient. However, it can only predict DTIs for drugs or proteins with existing interactions.
GIP	Interaction profile	Regularized least square	No need to calculate any similarity features for drugs or proteins.
GIP+WNN	Substructure; sequence; interaction profile	K nearest neighbour	Extension of GIP with the ability to predict for drug or protein without any interaction information.
PSL	Substructure, ligand, gene expression, side effect, ATC; sequence, PPI, GO;	Probabilistic graphical model	A framework where multiple features can be combined to achieve better performance, but with high time and space complexity.
RBM	Interaction profile; modes of action	Neural network	Take direct/indirect interaction and modes of actions into consideration.

## 2.4 Chapter summary

In this chapter, we introduced the related works in deep learning and protein analysis, respectively.

From the perspective of deep learning, we specifically introduced the three deep learning models, CNN, CNF and RNN, which have been applied to protein analysis. We also introduced the distributed representation in natural language processing, providing preliminary knowledge for different implementations of the distributed representation of protein sequences. Finally, we introduced in the deep learning implementation of transfer learning and multitask learning, demonstrating their capability in leveraging limited quantities of training data for rigorous model training. The different deep learning models and their applications to other areas such as natural language and image processing demonstrated the great potential of improved performance when they are applied to protein analysis.

From the perspective of protein analysis, we introduced the existing applications of deep learning and their limitations.

In terms of protein sequence representation, we categorised the existing representations of protein sequences into four categories and discussed the limitation of existing representations in their application to different granularities of prediction tasks. We also introduced an existing implementation of the distributed representation of protein sequences, *ProtVec*, which has the disadvantage of ignoring the order of amino acids in protein sequences. In this thesis, we introduce another three implementations in **Section 4.1.2** to take into consideration the order of amino acids in protein sequences. We also propose an application framework of the distributed representations of protein sequences in **Section 4.1.4**, demonstrating their flexibility in different granularities of prediction tasks.

In terms of deep learning model application, we introduced existing applications of CNN, CNF and RNN in protein analysis. We notice that the related researches were conducted in an *ad hoc* manner, with a lack of the systematic design for correlating multiple prediction tasks and the incorporation of biological background knowledge in the design of the deep learning models. In this thesis, we propose two multitask frameworks and a deep transfer learning framework in **Section 4.3** and **Section 4.4**, respectively, with the purpose of leveraging limited quantities of training data for rigorous model training by incorporating biological background knowledge and correlations between prediction tasks.

Finally, we introduced existing state-of-the-art machine learning methods for prediction tasks that are addressed in this thesis. We specifically demonstrated their rationales, input features and machine learning algorithms. The methods introduced in this chapter are later used as the compared methods in quantitative performance evaluations in **Chapter 5 – Chapter 8**.



## 3 Scientific method

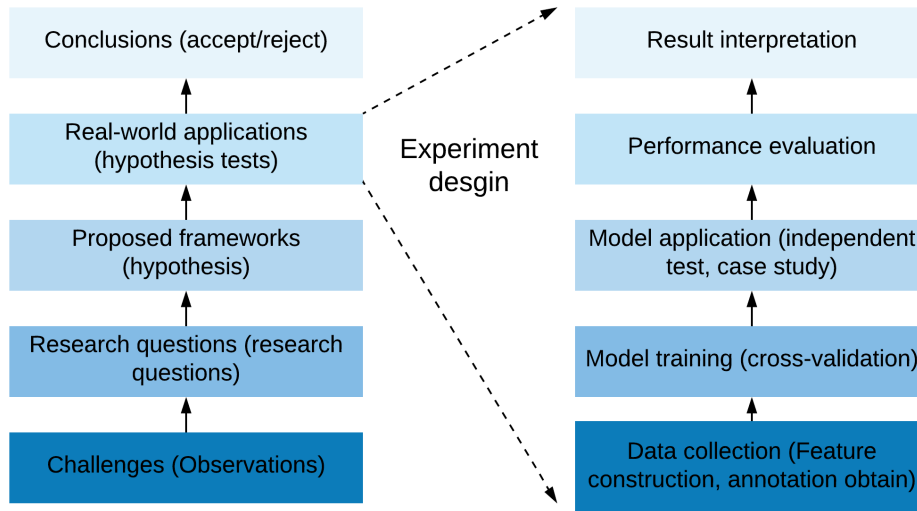
In this thesis, we propose deep learning frameworks for protein sequential, structural and functional analysis, for which comprehensive experiments are carefully designed to evaluate each of the proposed deep learning frameworks for specific prediction tasks. In this chapter, we introduce the general scientific methods that are used in rest of this thesis.

### 3.1 Scientific method in general

According to Garland, T. (2016), scientific method refers to an iterative and ongoing process of seven steps including making observations, forming research questions, formulating hypotheses, developing testable predictions, testing predictions, refining/altering/expanding/rejecting hypotheses and developing theories [257]. In this thesis, based on the existing challenges (observations) in applying deep learning to protein analysis, we raised three research questions, and for each research question, we proposed one or two deep learning frameworks for addressing corresponding challenges (hypotheses). We tested the effectiveness of these frameworks by applying them to different predictive tasks, whereby each of these applications employ a similar design of experiments that is widely adopted in machine learning applications [258].

According to Bradley, A.P. (1997), machine learning generates systems that learn by shown labelled examples [259]. It provides case studies in evaluating machine learning systems with five key components including the *data*, the *learning algorithms*, the *training methodology*, the *performance measures* and the *comparative techniques* [259]. Accordingly, in this thesis, the experimental design for machine learning application is composed of four aspects, including *data collection* (data), *model training and model application* (training strategy), *performance evaluation* (performance measurements), and *result interpretation*. The *learning algorithms* introduced in the Bradley, A.P. (1997)'s

methodology are introduced in respective chapters, and we added *result interpretation* to allow computational and biological insights and conclusions. **Figure 3.1** depicts an overview of the scientific method and the experimental design in this thesis.



**Figure 3.1** An overview of the scientific method that is used in this thesis.

## 3.2 Data Collection

Data collection refers to the process of gathering or measuring observed information of the target object. It is an important step for all sorts of research and the collected data can be qualitative or quantitative depending on the research method that is used.

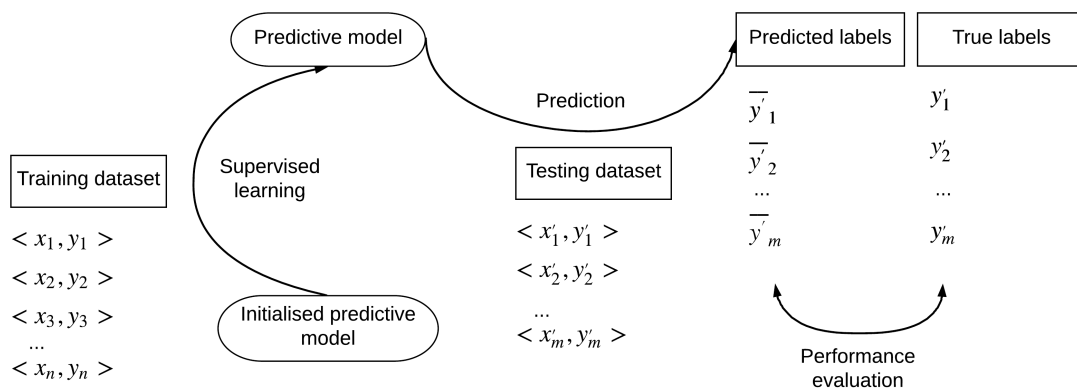
In machine learning, there are different requirements of data collection for supervised and unsupervised learning. *Supervised learning* infers a function from a set of labelled training examples. Therefore, each training example is composed of the input information, which may be relevant to the target output, and the output value (categorical or numerical), which indicates the true state of the target output. In many cases, the collection of the true output values is the most difficult step. Contrary to supervised learning, *unsupervised learning* seeks to infer the hidden function from unlabelled data.

The collection of unlabelled data is similar to the collection of labelled data except that it only requires the input information.

### 3.2.1 The construction of training dataset and testing dataset

In this thesis, we used supervised learning techniques for most of the applications of the proposed deep learning/machine learning frameworks. For each prediction task, to which the proposed deep learning/machine learning frameworks are applied, we collected data to construct a training dataset and a testing dataset.

Let  $x$  denotes the input information, and  $y$  the output, the *training dataset* is composed of pairs  $\langle x, y \rangle$  based on which the supervised learning process automatically learns the parameters of the predictive model. In order to evaluate the prediction performance of the learned predictive model, testing datasets are constructed. The *testing dataset* is also composed of pairs of input information and corresponding output values  $\langle x', y' \rangle$  that are unseen in the training dataset. Based on input information  $x'$ , the learned predictive model produces the predicted output value  $\bar{y}'$ , which is compared to the true output value  $y'$ . **Figure 3.2** demonstrates the roles of the training dataset and the testing dataset in the application of machine learning predictive models.



**Figure 3.2** The roles of training and testing datasets in the application of machine learning predictive models.

### 3.2.2 The collection of labelled data and the construction of input features

In **Chapter 5 – Chapter 8**, we propose the application of deep learning frameworks to five different prediction tasks, including protein family prediction, phosphorylation site prediction, drug-target interaction prediction, protein SSP prediction and IDP/IDR prediction. We collected the labelling data (output values) for each of these five prediction tasks from publicly available databases including the Pfam database [70], the Swiss-Prot/UniProt database [39] and the Phospho.ELM database [94], the KEGG database [96], the BMRB database [93], and the ModiDB database [92], respectively.

Since the five prediction tasks are all based on protein sequences, the input information for constructing training and testing datasets are collected mainly from the Swiss-Prot/UniProt database [39] (see **Section 1.2.4**), which represents one of the most complete protein sequence database. For each of the input protein sequences, we construct input features that are directly fed to the proposed deep learning/machine learning models. These input features so constructed from protein sequences are used to describe different aspects of a protein including protein homology profiles, physicochemical properties, structural information, sequential patterns and motifs, and distributed representations of protein sequences. We calculated the former four types of input features using existing bioinformatics tools such as PSI-BLAST [99], Pse-in-One [260], PSIPRED [103], DISOPRED3 [101], and local implementations of existing feature extraction algorithms. For detailed introduction of the data sources and bioinformatics tools, please refer to **Section 1.2.4** and **Section 1.2.5**. In **Section 4.1**, we also propose a distributed representation of protein sequences and its application in three different application scenarios.

## 3.3 Model training and model application

The step of model training refers to a defined automatic process that determines the parameters of predictive models from labelled training examples. The key component of

this step is the training algorithm, which usually consists of an objective function and a gradient descent process. The *objective function* of a mathematical program is what an optimization procedure uses to select better solutions over poorer solutions [261]. In machine learning, a typical objective function is defined as the difference between the target values that are predicted from the input feature vectors and the true target values that are originally collected. The supervised learning process accordingly adjusts the predictive model so that the difference between the predicted values and the true values is minimised.

The gradient descent is a first-order optimisation algorithm used to find the local minimum of the objective function [262]. It guides the adjustment of the parameters in machine learning models over multiple iterations, so that the output of the objective function gradually decreases over the time of training. The training of the model is completed when the output of the objective function or the number of iterations satisfies a certain condition.

Once the process of model training is completed, the pre-trained model is expected to be applied to new data for the purpose of prediction, classification, regression and clustering. This step represents the purpose of the machine learning study and is usually discussed in real-world application scenarios. However, before the model is applied to real-world scenarios, the pre-trained machine learning models are required to be evaluated in multiple rounds of tests, such as the cross-validation test and the independent test.

In this thesis, for the purpose of model training and model application, we use a variety of experimental techniques including cross-validation tests, independent tests, vector space analysis, correlation analysis, and case studies, to demonstrate the effectiveness and prediction performance of the proposed deep learning/machine learning frameworks. For detailed introduction of performance evaluation, please refer to the next section. In this section, we provide a brief introduction to each of the experimental techniques that are used in the following three chapters.

### 3.3.1 The cross-validation test

According to Gandhi, V. (2015), *cross-validation* is both an empirical and a heuristic approach typically carried out to assess how the results of a statistical analysis generalize to a set of independent data [263]. Therefore, the purpose of cross-validation is to, during the process of model training, define a validation dataset based on which the model is evaluated. In order to make the evaluation more robust to data variability, multiple iterations of training and validation are performed. In this thesis, we mainly used the  $n$ -fold cross-validation [264]. In  $n$ -fold cross-validation, the training dataset is randomly divided into  $n$  equally sized disjoint partitions. In each iteration,  $n-1$  partitions are combined for training the predictive model, and the remaining partition is used for evaluating the prediction performance of the trained model. This process is repeated  $n$  times and in each iteration a different partition is used for validation. The  $n$ -fold cross-validation makes sure that each example in the training dataset is used at least once for the purpose of validation. Since the prediction performance is evaluated  $n$  times based on  $n$  different subsets of the training dataset, the insights made from the cross-validation test about the generalisation of the model to independent data are more reliable. In practice, the cross-validation test is also widely used to tune the values of hyper-parameters<sup>16</sup>.

### 3.3.2 The independent test

The *independent test* is performed to evaluate the performance of a pre-trained machine learning models on a set of independent data. It is an imitation of the real-world application where the distribution of the data is unknown. Ideally, the training datasets and the independent datasets are constructed from different data sources. However, due

---

<sup>16</sup> Hyper-parameter is the parameter whose value is pre-defined before the process of model training. It is usually used to define the structure of the machine learning model, the distribution of a variable, or the specifications that are related to the training process.

to the lack of labelled data, it is not uncommon to divide a whole dataset into a training dataset and an independent dataset based on which the cross-validation test and the independent test are respectively conducted.

### 3.3.3 Vector space analysis

According to the previous section, the input information of a prediction task is usually represented as input feature vectors which are directly fed to machine learning models. These feature vectors form a feature vector space in which each possible input example can be mapped to a point. In this thesis, we conducted vector space analysis to evaluate the performance of the proposed feature extraction modules, by plotting the extracted vector representations of the input examples with respect to the labels of the examples. Assuming the module of feature extraction increases the quality of the input representation, the mapped distances between the examples annotated with the same label are expected to be smaller than those between the examples that are annotated with different labels. In **Section 5.4** and **Section 7.6.9**, we use the *principal component analysis (PCA)* [265] and the *t-distributed stochastic neighbour embedding (t-SNE)* technique [266] to map the multi-dimensional representations of protein sequences to 2-dimensional representations, so that the input examples with respect to their annotated labels can be manually visualised and validated.

### 3.3.4 Correlation analysis

Correlation analysis is performed to identify the strength of relationships between a pair of variables [267]. In this thesis, we conducted correlation analysis to investigate the factors that are closely related to the prediction performance of the pre-trained deep learning/machine learning models. Especially for models that achieve inferior performance than existing methods, we manually selected the factors that are likely to affect the prediction performance and analysed the prediction performance of the models in different sub datasets that have different levels of the investigated factors. For example,

in **Section 5.6.3**, we investigated the correlation between the performance of the pre-trained models and eight potential factors in protein family prediction, and in **Section 7.6.8**, we investigated the prediction performance of the pre-trained models in terms of the number of annotated phosphosites for different kinases.

### 3.3.5 Case study

The *case study* is a research strategy that focuses on a single case of a particular phenomenon [268]. In this thesis, in order to demonstrate the use case of proposed predictive models in real world applications, we applied the pre-trained predictive model to individual examples in specific prediction tasks. We mainly focus on the quantitative and qualitative analysis of multiple aspects including the overall performance of the predictive models, the potential reasons of making false positive predictions, existing experimental evidence of the true positive predictions, and the validation of the proposed hypothesis. In **Section 6.4.5**, we applied the pre-trained multitask model to protein p53 and SOSS1, from which we validated prediction performance of the pre-trained models for IDP/IDR prediction, demonstrated the positive effect of incorporating protein SSP prediction as the supporting task, and illustrated the real-world use case and the advantage of the proposed multitask model compared to existing single-task models.

## 3.4 Performance Evaluation

To evaluate the performance of a particular machine learning method, pre-trained machine learning models are applied to a set of independent data. By comparing the generated output values to true labelled values, different performance measurement scores are calculated and used as indexes to evaluate the performance of the pre-trained models.

In this thesis, we use different sets of evaluation measurements in different prediction tasks, including precision, sensitivity (recall), specificity, accuracy (ACC), the Matthews coefficient of correlation (MCC), the Pearson's coefficient of correlation (PCC or R), the



area under the ROC curve (AUC), and the area under the precision-recall curves (AUPR) [269]. Please refer to **Appendix A** for detailed information of these evaluation measurements. Since the output values of the proposed predictive models are probabilistic values ranging from 0 to 1, the calculation of the previous five measurement scores relies on the selection of the cut-off thresholds to determine the positive predictions from the negative ones. In **Section 5.5.3** and **Section 7.4.8**, considering the different ranges of output values that are produced by the compared machine learning methods, we used the false-positive rate (FPR) as the measurement for selecting cut-off thresholds. For example, a threshold of 0.0001 for the FPR score indicates that, in every 10,000 predictions, at most 1 false positive prediction is allowed. When the cut-off values for FPR scores are set, corresponding cut-off values of the output probabilities are automatically adjusted according to the output range of each predictive models, respectively. In **Section 6.2** and **Section 6.4**, a fixed threshold value of 0.5 for the output probability was used in IDP/IDR predictions, according to the evaluation methodology introduced in the CASP assessments [270, 271].

A second challenge of evaluating predictive models in protein analysis is to conduct evaluations based on imbalanced data. Imbalanced data is ubiquitous in protein analysis. For example, among thousands of protein families and hundreds of thousands of proteins, there are only tens of thousands of protein family annotations. Also, the negative examples in phosphorylation datasets is usually hundreds of times more than the positive examples. Therefore, the calculated evaluation scores in terms of sensitivity, specificity and accuracy do not reflect the true prediction performance of the evaluated models. Therefore, in this thesis, we introduce the use of the balanced accuracy (BACC), which is defined as follows [272],

$$BACC = \frac{1}{2} \left( \frac{TP}{TP+FN} + \frac{TN}{TN+FP} \right) \quad [\text{Eq. 3.1}]$$

where TP, TN, FP, FN represents the number of true positive predictions, true negative predictions, false positive predictions and false negative predictions, respectively. In this

way, the balanced accuracy is calculated as the average of the true positive rate and the true negative rate. Therefore, the overall performance is not over-estimated due to the high true negative rate. The AUC score is less robust to imbalanced dataset compared to AUPR. However, since the AUC score and the ROC curves are widely used in existing prediction methods, we used the latter two in most of the application tasks. MCC is another evaluation measurement that is designed to evaluate binary classifiers based on imbalanced datasets, but it relies on the selection of an appropriate cut-off thresholds.

### **3.5 Result interpretation**

Result are interpreted qualitatively or quantitatively in this thesis. The quantitative interpretation of results is usually straightforward. From the perspective of qualitative result interpretation, we focus our discussion in following aspects. Firstly, in comparison experiments, we are seeking to find out which methods perform better under which conditions. In practice, it is difficult for a single method to beat all other compared methods in terms of every evaluation measurements. Therefore, it crucial to discuss the top-ranked methods in terms of different evaluation measurements with different experimental conditions. Secondly, in failed experiments, we are seeking to analysis the potential factors that may affect the prediction performance of the proposed predictive models, based on which useful suggestions can be made about the application of the model in real-world scenarios. Thirdly, by combining the results of different sections and different chapters, we are seeking to conclude a set of heuristics for the future design of deep learning frameworks in protein analysis.

### **3.6 Chapter summary**

In this chapter, we introduced the five-step methodology of applying the proposed deep learning/machine learning frameworks to prediction tasks in protein analysis. In data collection, we demonstrated the construction of training and testing datasets from existing protein related data sources using various bioinformatics tools. In model training

and model testing, we introduced five experimental methods that are used in the rest of this thesis, including cross-validation tests, independent tests, vector space analysis, correlation analysis and case study. In performance evaluation, we demonstrated two of the unique challenges in evaluating predictive models in protein analysis, *i.e.* the cut-off threshold selection and the evaluation of imbalanced datasets. Finally, we introduced the step of results interpretation which produces sectional insights based on the experimental results. In this thesis, scientific methods introduced above are applied to various aspects of the practical experimental design, constituting the main contents of the application of proposed deep learning/machine learning frameworks to respective prediction tasks.

## 4 Deep learning frameworks for protein analysis

Proteins represent one of the main functional gene product of gene expression. They play important functions in various biological processes including enzyme catalysis, DNA binding and molecular transport. Protein functions are determined by protein three-dimensional structures which are folded from amino acid sequences. Due to the different rates of increase in protein sequence data and protein structural and functional data, computational methods for predicting protein structures and functions from protein sequences are needed.

In this chapter, we introduce four deep learning frameworks that are designed to address the three research questions raised in **Section 1.4**. From the perspective of input representation, we propose the distributed representation of protein sequences as a source of potential complementary input features. From the perspective of machine learning algorithms, we propose the general deep learning framework, the multitask deep learning framework, and the deep transfer learning framework for predicting protein structural and functional properties based on protein sequences.

### 4.1 The distributed representation of protein sequences

To predict protein structures and functions from protein sequences, the first and most fundamental step is to represent protein sequences into machine understandable data. In this section, we introduce the distributed representation of protein sequences, namely prot2vec, and its application to three granularities of prediction tasks including whole sequence-level, residue-level and biological interaction-level predictions.

#### 4.1.1 Notation system for protein sequences

Most protein sequences are composed of the 20 naturally occurring amino acids [273]. They were first notated as sequences of 3-letter symbols, which caused application difficulties due to the spacing in long protein sequences [42]. In 1958, Gamov and Ycas for the first time introduced the use of 1-letter symbols for denoting amino acid sequences [274], which was later formalised by IUPAC-IUB Commission on Biochemical Nomenclature in 1971 [42]. This 1-letter denotation system greatly simplified the protein sequence alignment. The 1-letter notations and 3-letter notations for protein sequences are shown in **Table 4.1**.

**Table 4.1** The 1-letter and 2-letter denotations for amino acids (AA).

AA	3-letter	1-letter
Alanine	Ala	A
Arginine	Arg	R
Asparagine	Asn	N
Aspartic acid	Asp	D
Cysteine	Cys	C
Glutamic acid	Glu	E
Glutamine	Gln	Q
Glycine	Gly	G
Histidine	His	H
Isoleucine	Ile	I

AA	3-letter	1-letter
Leucine	Leu	L
Lysine	Lys	K
Methionine	Met	M
Phenylalanine	Phe	F
Proline	Pro	P
Serine	Ser	S
Threonine	Thr	T
Tryptophan	Trp	W
Tyrosine	Tyr	Y
Valine	Val	V

\* Any amino acids other than the above 20 amino acids are usually denoted as the 1-letter symbol **X**.

Based on the 1-letter notations, the FASTA format was released with the FASTA suite [275], a protein sequence alignment tool that was developed in 1985. This format allows the storage, transform and analysis of multiple protein sequences at the same time. For each protein, the FASTA format consists of a description line starting with the

'>' symbol, and one or more lines of one-letter notation of the protein sequence<sup>17</sup>. The description line contains the identification of the protein and optionally other information of the protein. An example [39] of the FASTA format is shown as follows,

```
>sp|Q196Z8|062L_IIV3 Uncharacterized protein 062L OS=Invertebrate iridescent virus 3 GN=IIV3-062L
PE=4 SV=1
MNPEGENELQVYLRDGDRLVMALGLRHGLRRGLRRGLRRGLRRTGDLRDGDLRLDGDRLD
GDLDLLDGDRLRDGDLRLDGDLDLLEGLRDGDLDLLDGDHLRDGDVDRLEGDLRDGDLRL
DGDFLVLPHTTAWVFLPNSFNFLVQTLFAPFSYWLHNGIVFINSKILFIKSPQLPSYF
AMALRQCKSSWLGHSTPANRFCIVVQINDQLGLKNSINSRRSALWSGGTASC

>sp|O55706|062L_IIV6 Uncharacterized protein 062L OS=Invertebrate iridescent virus 6 GN=IIV6-062L
PE=3 SV=1
```

In this thesis, we analyse protein sequences based on its 1-letter notations and develop protein sequence-based prediction web services using the FASTA format.

#### 4.1.2 Prot2vec: the distributed representation of protein sequences

Amino acid sequences, from the moment they were notated as sequences of one-letter alphabets, form a biological language that is capable of encoding the information of proteins. Ganapathiraju, M., et al. (2005) has proposed the analogy between biology and natural languages, where genome sequences were mapped to raw texts from documents; protein expression, structures and functions were mapped to meaning; and biological interactions were mapped to the topics of the documents [163] (see **Figure 2.4**). The natural language processing technique, *n*-gram analysis, was comparatively performed for biological sequences [276].

In recent years, the development of word embedding [164] has greatly forwarded the research of natural language processing, especially for some of the most challenging tasks such as document summarization [277], syntactic parsing [278] and sentiment

---

<sup>17</sup> FASTA format:

[https://blast.ncbi.nlm.nih.gov/Blast.cgi?CMD=Web&PAGE\\_TYPE=BlastDocs&DOC\\_TYPE=BlastHelp](https://blast.ncbi.nlm.nih.gov/Blast.cgi?CMD=Web&PAGE_TYPE=BlastDocs&DOC_TYPE=BlastHelp)

analysis [279]. The basic idea of distributed representation in word embedding (see **Section 2.1.3**) is that condensed vector representations of words (, sentences and documents) can be extracted from the hidden layers of the neural network trained for specific natural language processing tasks [280, 281]. Such representation is extracted based purely on the raw texts using unsupervised learning techniques and has the merits of being semantic-relevant (see **Section 2.1.3**), condensed and fixed-length when compared to existing amino acid-based representations.

More recently, Asgari, E. and Mofrad, M.R.K. (2015) [193] proposed the distributed representation of biological sequences, namely ProtVec, based on the word2vec model (see **Section 2.1.3**). The basic idea of ProtVec is that the distributed representation of a protein sequence can be obtained by summing up the distributed representations of its biological words. In this thesis, we rename the ProtVec to  $\text{prot2vec}^{\text{add}}$ , considering its generation strategy based on vector summation. The disadvantage of this implementation strategy is that it ignores the order of biological words (or amino acids) in protein sequences.

In this section, in order to overcome the disadvantage of the summation strategy of  $\text{prot2vec}^{\text{add}}$ , we introduce the distributed representation of protein sequence,  $\text{prot2vec}$ , with the inspiration of two of the most successfully applied word embedding representations, word2vec [170] and doc2vec [169]. In addition to the summation implementation  $\text{prot2vec}^{\text{add}}$  introduced by Asgari, E. and Mofrad, M.R.K. (2015), we propose three new implementations of the  $\text{prot2vec}$ , namely  $\text{prot2vec}^{\text{non-overlap}}$ ,  $\text{prot2vec}^{\text{fetch}}$ , and  $\text{prot2vec}^{\text{inference}}$ , and provide an overall qualitative comparison among the four implementations.

**$n$ -grams of protein sequences.** The  $n$ -gram model was proposed to analyse sequential data in natural language processing tasks [282-284].

**Definition 4.1** In an  $n$ -gram language models, two histories end in the same  $n-1$  words are considered equivalent [282].

Let  $w_1^N$  represents the word sequence  $w_1, w_2, \dots, w_N$ ,  $Pr(w_1^N)$  represents the probability of this string of words, and  $k > n$ , the above definition is represented as follows [282],

$$Pr(w_k | w_1^{k-1}) = Pr(w_k | w_{k-n+1}^{k-1}) \quad [\text{Eq. 4.1}]$$

Therefore, in  $n$ -gram language model, for a word sequence of  $n$  words, the unigram ( $n = 1$ ) model allows  $Pr(w_1^k) = Pr(w_1) \times Pr(w_2) \dots \times Pr(w_k)$ , bigram ( $n = 2$ ) model allows  $Pr(w_1^k) = Pr(w_1 | w_0) \times Pr(w_2 | w_1) \dots \times Pr(w_k | w_{k-1})$ , and trigram ( $n = 3$ ) model allows  $Pr(w_1^k) = Pr(w_1 | w_{-1}^0) \times Pr(w_2 | w_0^1) \times Pr(w_3 | w_1^2) \dots \times Pr(w_k | w_{k-3+1}^{k-1})$ , etc.

Protein sequences are continuous chains of amino acids that are similar to natural language sentences which do not have word splitters such as spaces and punctuations. Typical languages that do not contain word splitters include Chinese, Japanese, Thai and Korean [285]. Due to the lack of punctuations and spaces, sentences in these languages are usually split into  $n$ -grams at the character level, which are later recognised as *tokens* (words) or *non-tokens*. This additional step of tokenisation makes it more challenging to perform prediction tasks for such languages. With protein sequences, similar techniques were used to split sequences of amino acids into short segments of specific lengths [193]. These segments are analogous to words in natural languages and we refer to them as the biological words in this thesis.

To split protein sequences into  $n$ -gram biological words, the most intuitive way is to regard each residue as one ‘word’, which is equivalent to the case of unigram. However, a more general way of dealing with biological sequences is to split them into overlapping or non-overlapping  $n$ -grams [16]. **Figure 4.1** demonstrates the three ways of splitting protein sequences, including unigram ( $n = 1$ ), overlapping  $n$ -gram ( $n = 3$ ) and non-overlapping  $n$ -gram ( $n = 3$ ).

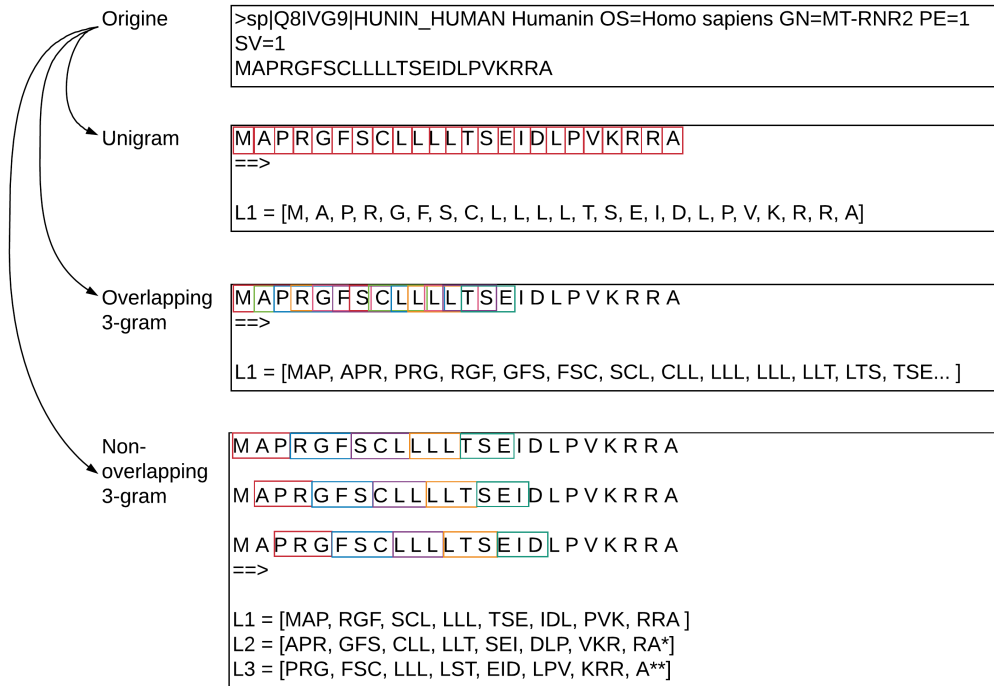
- Overlapping splitting. A sliding window of size  $n$  is used to cut a protein sequence

$S = r_1 r_2 \dots r_{|S|}$  of length  $|S|$  into  $(|S| - (n - 1))$  sequence segments, where the



sliding window of the  $i$ -th segment starts from the  $i$ -th residue of the sequence. Here, each sequence segment corresponds to an  $n$ -gram biological word. Note that, in this case, one protein sequence is split into only one list of  $n$ -grams, denoted as  $l_W = [w_1, w_2, \dots, w_{|S|-(n-1)}]$ .

- Non-overlapping splitting. A protein sequence  $S = r_1 r_2 \dots r_{|S|}$  of length  $|S|$  is split into  $n$  lists of  $n$ -gram biological words. For the  $j$ -th list, a sliding window of size  $n$  is used to cut the sequence  $r_j r_{j+1} \dots r_{|S|}$  of length  $L = |S| - j + 1$  ( $j = 1 \dots n$ ) into  $L \bmod n$ , where the modulo calculus mod is used to calculate the remainder of a division calculation, sequence segments, where the sliding window of the  $i$ -th segment starts from the  $j \times n$ -th residue of the sequence. Therefore, the  $j$ -th list of biological words is denoted as  $l_W = [w_1, w_2, \dots, w_{(|S|-j+1) \bmod n}]$



**Figure 4.1** Unigram and 3-gram splitting with overlapping and non-overlapping strategies for the amino acid sequence of protein HUNIN\_HUMAN.

**Generating the distributed representation of protein sequences.** Different from natural languages, protein sequences are composed of chains of amino acids with no observable rules or grammars. However, sequential motifs have been studied and proposed for proteins with specific functions [68, 286, 287]. The database PROSITE [68] has listed the curated sequential motifs that are shared by proteins within the same protein families. For example, the active binding site (D) of the Serine/Threonine kinases have a local sequential motif of [LIVMFYC]-x-[HY]-x-D-[LIVMFY]-K-x(2)-N-[LIVMFYCT](3) [288], when represented as the regular expression<sup>18</sup>. It indicates that there exists local dependency among neighbouring residues within protein sequences.

In natural language processing, the local dependency among neighbouring words and phrases is one of the main sources of information that are used for sequential modelling [280]. For example, an  $n$ -gram model is used to predict the  $i$ -th word based on its previous  $n$  words, assuming that the determination of the  $i$ -th word only depends on the  $n$  neighbouring words that are in the upstream of the word sequences [282]. Based on this idea, Mikolov, T., et al. (2013) proposed the skip-gram model, which generates word representations (word2vec) that are useful for predicting neighbouring words in sentences or documents [289]. Please refer to **Section 2.1.3** for the detailed introduction of the word2vec and doc2vec models.

Considering the local dependency among neighbouring residues in protein sequences, it is straightforward to apply the skip-gram model to generate vector representation for biological words. With the inspiration of word2vec, Asgari, E. and M. R. K. Mofrad (2015) proposed the continuous distributed representation of biological sequences, generating the distributed representation of a protein sequence as the sum of the word2vec of all its biological words [193]. However, the summation strategy ignores the order of biological words which implicitly represents the order of amino acids in protein sequences.

---

<sup>18</sup> Regular expression refers to a sequence of characters defined as the pattern for searching purposes.

Inspired by the distributed representation of sentences and documents, doc2vec, we propose the following steps to obtain the distributed representation of protein sequences.

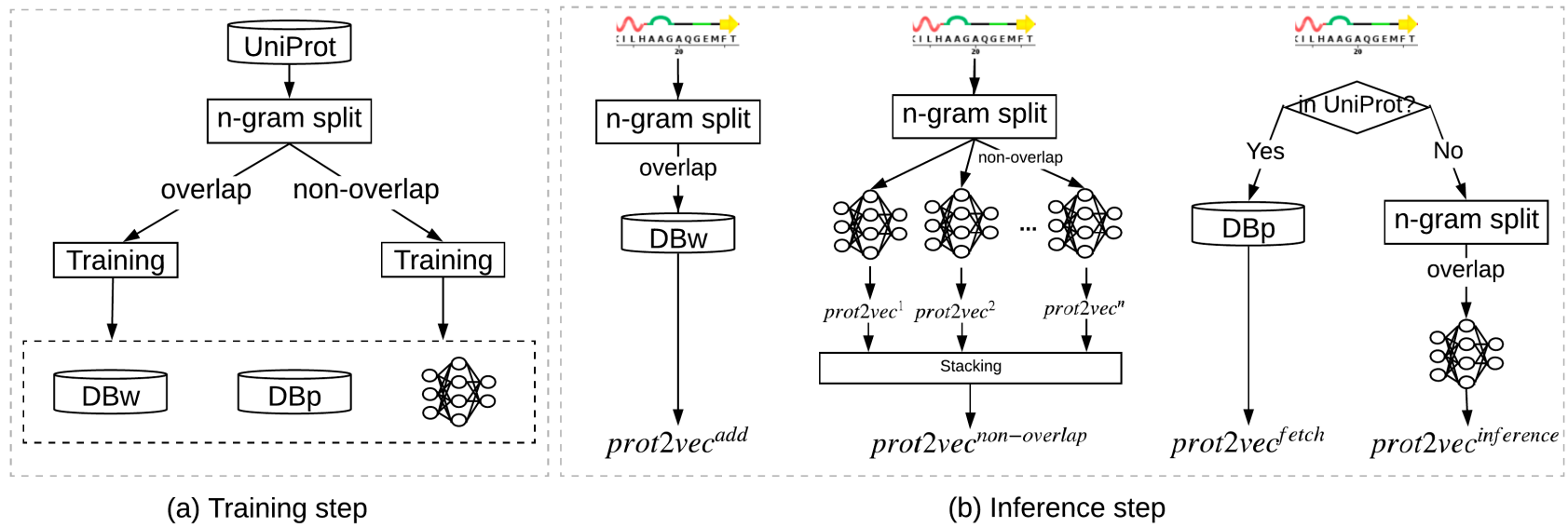
1. We constructed the training dataset from a large protein sequence database for training the model to generate distributed representation of protein sequences. In total, 551,704 reviewed protein sequences from the Swiss-Prot/UniProt database [39] were downloaded in Aug 2016 and extracted;
2. For each protein sequence in the dataset, we performed an overlapping  $n$ -gram split which produces a list of  $n$ -gram biological words [193]. Asgari and Mofrad showed that 3-gram split led to the best performance [193]; we thus produced all the 3-gram biological words for all the extracted 551,704 protein sequences based on 3-gram split;
3. We fed 3-gram biological words to the doc2vec algorithm for training a vector generation model, resulting in a trained distributed memory network  $\mathcal{N}$  for future inference, a database  $DB_w$  that mapped the biological words to their pre-trained vectors, and a database  $DB_p$  that mapped the protein sequences to their pre-trained vectors. The doc2vec algorithm [170] in the Gensim<sup>19</sup> package was used for this training process;
4. In the inference step, for any target protein sequence or sequence segment, the distributed representation of the target protein sequence or sequence segment was then generated in four ways:
  - a.  $\text{prot2vec}^{\text{add}}$ . The protein sequence  $S$  was split into overlapping  $n$ -gram biological words. The representations of all biological words were then directly fetched from the database  $DB_w$  and summed up to form the representation of protein sequence  $S$ . This representation was proposed by Asgari, E. and Mofrad, M.R.K. (2015) [193] and named as ProtVec.

---

<sup>19</sup> The Gensim package: <https://radimrehurek.com/gensim/>

- b.  $\text{prot2vec}^{\text{non-overlap}}$ . The protein sequence  $S$  was split into  $n$  lists of non-overlapping  $n$ -gram biological words. The biological word lists were then fed to the pre-trained model  $\mathcal{N}$ , which generated a vector representation for each of the  $n$  lists of biological words. The final representation of the protein sequence  $S$  was constructed by stacking the  $n$  generated  $\text{prot2vecs}$ .
- c.  $\text{prot2vec}^{\text{fetch}}$ . The protein sequence  $S$  was searched against the training databased, such as UniProt. If it existed in the database, its vector representation was fetched directly from the database  $DB_p$ . If not,  $\text{prot2vec}^{\text{inference}}$  was generated. Different from natural language processing where a corpus can hardly include all possible sentences, the recognition speed of new protein sequences is relatively slow and the protein sequence database Swiss-Prot/UniProt is considered to contain most of the functional proteins that may appear in a protein sequence-based analysis. Therefore,  $\text{prot2vec}^{\text{fetch}}$  is listed as a practical implementation of the  $\text{prot2vec}$ .
- d.  $\text{prot2vec}^{\text{inference}}$ . The protein sequence  $S$  was split into a list of overlapping  $n$ -gram biological words. The list of biological words was later fed into the pre-trained generative model  $\mathcal{N}$ , which generates the vector representation of  $S$  via inference. In this step the  $\text{doc2vec}$

In this thesis, the general technique of the distributed representation of protein sequences is referred to as  $\text{prot2vec}$ , which can be implemented using the four strategies in step 4, *i.e.*  $\text{prot2vec}^{\text{add}}$ ,  $\text{prot2vec}^{\text{non-overlap}}$ ,  $\text{prot2vec}^{\text{fetch}}$  and  $\text{prot2vec}^{\text{inference}}$ . **Figure 4.2** demonstrates the generation of  $\text{prot2vec}$  in terms of the training step and the inference step.



**Figure 4.2** Difference implementations of prot2vec in the training and inference step.

### Qualitative comparison among the four implementations of prot2vec.

Combining the two strategies of splitting protein sequences and the two ways of generating distributed representations of protein sequences, **Table 4.2** summarises the different implementations of prot2vecs with their advantages and disadvantages. Among the four different implementations,  $\text{prot2vec}^{\text{fetch}}$ ,  $\text{prot2vec}^{\text{non-overlap}}$  and  $\text{prot2vec}^{\text{inference}}$  are doc2vec-based. Therefore, the generated representations take into consideration the order of amino acids. On the other hand,  $\text{prot2vec}^{\text{fetch}}$  and  $\text{prot2vec}^{\text{add}}$  are constructed by fetching representations directly from database  $DB_p$  and  $DB_w$ , respectively. Therefore, the respective time complexity for these two implementations are within  $O(1)$  and  $O(n)$ <sup>20</sup>. Note that,  $\text{prot2vec}^{\text{add}}$ ,  $\text{prot2vec}^{\text{non-overlap}}$  and  $\text{prot2vec}^{\text{inference}}$  can be applied to any protein sequence segments that are longer than 3 residues, while  $\text{prot2vec}^{\text{fetch}}$  is only available for whole protein sequences.

**Table 4.2** Different distributed representation of protein sequences.

	Overlapping $n$ -gram		Non-overlapping $n$ -gram	
	<i>Fetch</i>	<i>Inference</i>	<i>Addition</i>	<i>Inference</i>
Name	$\text{prot2vec}^{\text{fetch}}$	$\text{prot2vec}^{\text{inference}}$	$\text{prot2vec}^{\text{add}}$	$\text{prot2vec}^{\text{non-overlap}}$
Pros	Considers the order of biological words; with time complexity $O(1)$ .	Considers the order of biological words.	The calculation is of time complexity $O(n)$ .	Considers the order of biological words.
Cons	Only works for protein sequences in the training set.	Slow inference.	Ignores the order or biological words.	Requires extra calculation to combine representations of $n$ lists of biological words.

<sup>20</sup> The time complexity for  $\text{prot2vec}^{\text{add}}$  is  $O(n)$  because, for a sequence of  $n$  words, it requires  $n$  times representation fetch from the pre-trained mapping database  $DB_w$ .

### 4.1.3 The training of the prot2vec generative models

**Hyper-parameter tuning.** The generative model of the distributed representation is constructed based on feed-forward neural networks, whose hidden layers are trained to generate the vector representation. The training of the neural network  $\mathcal{N}$  was based on the tuning of multiple hyper-parameters. Detailed specifications of the hyper-parameters can be found in the latest release of the Gensim package. In this study, we specifically tuned the following hyper-parameters,

- Vector size ( $H$ ). This parameter specifies the size of the hidden layer and therefore the size of the generated vector representation.
- Window size ( $\omega$ ). This parameter determines the number of neighbouring words that are predicted from the target biological word. Larger size indicates a broader context from which the target word is modelled, whereas a smaller window size enforces a stronger focus on the local context.
- Learning rate ( $\partial$ ). This parameter determines the initial learning rate of the training process. A learning rate set too small may cause over-fitting.
- Training epochs ( $m$ ). This parameter specifies the number of training iterations that were performed on the training sequences. Usually, better performance can be achieved with more training epochs. However, overly sized training epochs may cause over-fitting.
- Inference epochs ( $n$ ). This parameter specifies the number of steps of inference that is performed to generate the vector representation of unseen sequences. It is only available for the inference step.

The model was trained on a machine with 8-core Intel(R) Core(TM) i7-4770 CPU and 16G of RAM.

**Performance evaluation.** In this thesis, the quality and effectiveness of prot2vec was validated respectively in two experimental settings. Firstly, we validated the legitimacy of the training step (**Figure 4.2 (a)**) by evaluating the accuracy of the inference step (**Figure 4.2 (b)**). Assuming the pre-trained prot2vecs in  $DB_p$  are accurate distributed representations of protein sequences in the training set, denoted as  $v_p$ , we performed an inference step to a subset (valid set) of the training sequences using the trained generative model  $\mathcal{N}$ , which generates the inferred prot2vecs of protein sequences in the valid set, denoted as  $\tilde{v}_p$ . Therefore, the accuracy of the inference step can be evaluated by the similarity or distance between  $v_p$  and  $\tilde{v}_p$ , which is defined as follows,

$$Sim(v_p, \tilde{v}_p) = \frac{1}{m} \sum_{j=0}^m cos(v_{p_j}, \tilde{v}_{p_j}) \quad [\text{Eq. 4.2}]$$

where  $m$  represents the size of the valid set, and  $cos$  refers to the cosine similarity<sup>21</sup> [290] between two vector representations. Here, larger  $Sim(v_p, \tilde{v}_p)$  is regarded as an indication of a superior generative model for prot2vecs. This evaluation is only applicable for inference-based distributions, such as prot2vec<sup>inference</sup> and prot2vec<sup>non-overlap</sup>. Secondly, we validated the effectiveness of the distributed representation of protein sequences by applying them to different protein sequence-based prediction tasks. Please refer to **Chapter 5, Section 7.4 and Chapter 8** for detailed results and analysis of the three prediction-task based validations. In this section, we focus on the validation of the legitimacy of the training step.

**Results and analysis.** We tuned the aforementioned hyper-parameters and selected the best performing generative models for prot2vec<sup>inference</sup> and prot2vec<sup>non-overlap</sup>, respectively, in terms of  $Sim(v_p, \tilde{v}_p)$  on the valid set. Note that prot2vec<sup>add</sup> and prot2vec<sup>fetch</sup> are constructed directed from the mapped distributed representations saved

---

<sup>21</sup> Cosine similarity is used to measure the similarity between two non-zero vector in terms of the cosine of the angle between them.



in  $DB_p$  and  $DB_w$ . Therefore, they do not involve actual approximate inference. As a result, we did not include them in the performance evaluation.

Among the 551,704 reviewed protein sequences that were used for training, we randomly selected 2,000 sequences as the valid set. **Table 4.3** listed the resulting score of  $\text{Sim}(v_p, \tilde{v}_p)$  with respect to corresponding vector size  $H$ , training epochs  $m$ , learning rate  $\partial$ , inference epochs  $n$ , and the window size  $\omega$ . We started the parameter setting from  $H=100$ ,  $m=300$ ,  $\partial=0.025$ ,  $n=20000$ , and  $\omega=25$ , and then manually performed grid search by changing one or two parameters at a time, to improve the cosine similarity between the pre-trained vector  $v_p$  and the inferred vector  $\tilde{v}_p$ .

**Table 4.3** The hyper-parameter tuning for the generative models of  $\text{prot2vec}^{\text{inference}}$  and  $\text{prot2vec}^{\text{non-overlap}}$ .

	$H$	$m$	$\partial$	$n$	$\omega$	Avg. Sim
$\text{prot2vec}^{\text{non-overlap}}$						
#1	<u>100</u>	<u>300</u>	<u>0.025</u>	<u>20000</u>	<u>25</u>	<u>0.810</u>
#2	-	-	-	500	-	0.330
#3	-	-	-	25000	-	0.805
#4	-	-	-	30000	-	0.800
#5	-	-	0.01	-	-	0.813
#6	300	-	-	-	30	0.744
#7	-	400	0.005	30000	-	<b>0.967</b>
$\text{prot2vec}^{\text{inference}}$						
#1'	<u>100</u>	<u>100</u>	<u>0.025</u>	<u>10000</u>	<u>25</u>	<u>0.863</u>
#2'	-	300	0.01	20000	-	0.950
#3'	-	400	0.005	30000	-	<b>0.974</b>

\* Underlines indicate that this particular hyper-parameter is set to the same value as that of the first set of hyper-parameters that were highlighted with underlines for  $\text{prot2vec}^{\text{non-overlap}}$  and  $\text{prot2vec}^{\text{inference}}$ , respectively.

Results #1~#4 demonstrate that  $\text{Sim}(v_p, \tilde{v}_p)$  was improved from 0.330 to 0.810 when the inference epochs were improved from 500 to 20,000 and decreased slightly to 0.805 and 0.800 when the inference step was further increased to 25,000 and 30,000. It indicates that the number of inference steps is positively correlated with the performance of the generative models, but there exists a limit after which more inference step may cause the decrease of the performance. Combining the results of #1, #4 and #7, we found that the limit of the number of inference steps is related to the number of training epochs. More specifically, despite the decrease of  $\text{Sim}(v_p, \tilde{v}_p)$  from 0.810 to 0.800 when the inference steps were increased from 20,000 to 30,000, the  $\text{Sim}(v_p, \tilde{v}_p)$  was improved to 0.967 when the number of inference steps was increased to 30,000, and the training epochs were increased to 400. Therefore, the increase of training epochs and inference steps should be performed together.

We also observed slight improvement in  $\text{Sim}(v_p, \tilde{v}_p)$  from 0.810 to 0.813 when the initial learning rate was decreased from 0.01 and 0.025. Note that larger training epochs meant that the model was repeatedly trained on the same training data more times, for which a smaller learning rate enables more sustainable training, despite the application of learning rate decay. Therefore, we decreased the initial learning rate with the increased of the training epochs. Finally, we changed the vector size from 100 to 300 and the window size from 25 to 30 in result #6, where the  $\text{Sim}(v_p, \tilde{v}_p)$  was decreased to 0.744 compared with 0.810 in result #1.

Here, larger vector sizes specify less condensed representations of the input sequence, whereas a larger window size involves more words in the contexts from which the distributed representation was generated. Based on the hyper-parameter tuning of  $\text{prot2vec}^{\text{non-overlap}}$ , we simplified the hyper-parameter tuning of the  $\text{prot2vec}^{\text{inference}}$ . Results #1'~#3' demonstrated that the best hyper-parameter settings for  $\text{prot2vec}^{\text{inference}}$  are  $H=100$ ,  $m=400$ ,  $\partial=0.005$ ,  $n=30000$ , and  $\omega=25$ , where  $\text{Sim}(v_p, \tilde{v}_p)=0.974$ . In comparison,

with the same hyper-parameter setting, the model of  $\text{prot2vec}^{\text{non-overlap}}$  achieved a  $\text{Sim}(v_p, \tilde{v}_p)$  of 0.967.

#### 4.1.4 The application framework of prot2vecs

Compared to biological representations including homology profiles, physicochemical properties, and structural properties,  $\text{prot2vecs}$  represent protein sequences in terms of their sequential patterns that are distributed in large protein sequence databases. Its generation model is trained based on unsupervised learning, and the generation of  $\text{prot2vecs}$  itself is fully automatic with no manual intervention. Therefore, the distributed representation of protein sequences can be regarded as a pure data-driven interpretation of amino acid sequences that is complementary to existing biological representations that are generated with biological heuristics.

In this section, we propose the application of  $\text{prot2vec}$  in the three granularities of prediction tasks at the whole-sequence level, residue-level and biological interaction-level, respectively. For each of the three application scenarios, we introduce the application context, quantitative input, expected output, symbolic definition, and applicable prediction tasks.

**Whole-sequence level predictions.** For protein structural and functional properties that are assigned per sequence, such as the protein family, the distributed representation of protein sequences,  $\text{prot2vec}$ , can be used directly as the feature vector for corresponding prediction models. The purpose of these whole-sequence level predictions is to computationally assign a categorical or numeric value for each protein, given its amino acid sequence as the raw input. Let  $M$  denote the model that is constructed to perform the prediction task,  $c_i \in \mathcal{C}$  the  $i$ -th category that a protein can be assigned to, and  $v_S$  the generated  $\text{prot2vec}$  of the input protein sequence  $S$  according to **Section 4.1.3**.

Whole sequence-level predictions based on the distributed representation of protein sequences can be formalised as follows,

$$M_{c_i}: f(v_S) \rightarrow \begin{cases} 1 & \text{if } S \in c_i \\ 0 & \text{if } S \notin c_i \end{cases} \quad [\text{Eq. 4.3}]$$

In this thesis, the prediction task  $M_{c_i}$  is modelled using machine learning models that are automatically learnt from constructed training datasets. The most widely used machine learning models include the support vector machines, neural networks, random forests, K-nearest neighbours and probabilistic graphical models.

Whole sequence-level predictions represent the most straightforward application of prot2vec. Technically, they require no extra pre-processing or post-processing of the vector representation and can be directly used as the input feature vector for most of the aforementioned machine learning models. Biologically, prot2vec is generated as a high abstractive representation of the sequential pattern of protein. It can be complementarily used in combination with the existing biological representations, simply via vector concatenation.

**Residue-level predictions.** For protein structural and functional properties that are assigned per residue, for example, protein secondary structures and intrinsic disorder, we propose the distributed representation of protein segments, termed context2vec, as the contextual feature vector for encoding local sequential pattern of the target residue. Here, the context2vec of any target residue  $r_j$  is generated for its local sequence segment  $w_j = r_{j-m}, r_{j-m+1}, \dots, r_j, \dots, r_{j+m-1}, r_{j+m}$ , where  $m$  denotes the number of neighbouring residues that are extracted from its upstream and downstream contexts. A sliding window of size  $L = 2 \times m + 1$  is used to extract the local sequence segment for each target residue. The context2vec<sup>add</sup> and context2vec<sup>inference</sup> of  $r_i$  are respectively generated for sequence segment  $w_j$  the same way prot2vec<sup>add</sup> and prot2vec<sup>inference</sup> are generated for the whole protein sequence  $S$ .

The purpose of residue level predictions is to assign a categorical or numeric value to each residue in a protein sequence, given the protein sequence  $S$  as the raw input data. Let  $M$  denotes the model that is constructed to perform the prediction task,  $c_i \in \mathcal{C}$  where  $\mathcal{C}$  is the set of allowable classes, and  $v_{r_j}$  the concatenation of the generated context2vec and various residue level biological representations, for the  $j$ -th residue  $r_j$  in the input protein sequence  $S$ . Residue-level predictions based on the distributed representation of local sequential contexts can generally be formalised as follows,

$$M_{c_i}: f(v_{r_j}) \rightarrow \begin{cases} 1 & \dots \text{if } r_j \in c_i \\ 0 & \dots \text{if } r_j \notin c_i \end{cases} \quad [\text{Eq. 4.4}]$$

Similar to whole-sequence level prediction, the task  $M_{c_i}$  is modelled with machine learning models.

Residue-level predictions are ubiquitous among protein sequence-based predictions, especially for protein structural properties. For each residue in the primary structure, corresponding structural and functional properties are computationally assigned to structurally characterise proteins. The structural properties that are predicted from protein sequences include protein secondary structures, secondary structure populations, solvent accessibility, and intrinsic order, while functional properties include phosphorylation sites, cleavage sites, and protein domains. An example of the input ‘seq’ and the output ‘SS’ of the protein secondary structure prediction tool PSSpred [291] for protein *p53\_human* [39] is shown as follows,

seq: 1	MEE <b>PS</b> VEPPLSQETFSDLWKLLPENNVLSPLSQAMDDLMLSPDDIEQWFTEDPGP	60
SS:	CCCCC <b>C</b> CCCCCCCCCHHHHHHHHHCCCCCCCCCCCCCHHHHHCCCCCHHHHCCCCC	
conf:	98666666789988146999998689988766777887567522687420221346775	
seq: 61	DEAPRMPEAAPVAPAPAAPTPAAPAPAPSWPLSSVPSQKTYQGSYGFRGLHSGTAK	120
SS:	CC	
conf:	322367766777777777767778888888888888876788888641499982488766	
seq: 121	SVTCTYSPALNKMFCQLAKTCPVQLWVDSTPPPGRVVRAMAIYKQSQHMTVEVRRCPHHE	180
SS:	CCCEEECCCCCEEEECCEEEEEEECCCCCCCCCEEEEEEECCCHHHCCHHCCCCC	
conf:	532032231286997258054499998169999928999998638111002132251545	
seq: 181	RCSDSDGLAPPQHLIRVEGNLRVEYLDNRNTRFRHSVVVPYEPPEVGSDDCTTIHNYMCNS	240
SS:	CCCCCCCCCCCCCEEEECCEEEECCEEEECCEEEECCEEEECCEEEECCEEEECCE	
conf:	578888777652499982899538863489854899974588887864068989985136	
seq: 241	SCMGGMNRRPILTIITLEDSSGNLLGRNSFEVRVCACPRDRRTEENLRKKGEPHHELP	300
SS:	CCCCCCCCCEEEECCEEEECCEEEECCEEEECCEEEECCHHHHHHHHHCCCCCCCC	
conf:	777888765358999983799967858899999937842210467886320145666789	

For these residue level predictions, one of the underlying assumptions is that the assignment of structural and functional properties of the target residue  $r_j$  is correlated to its local sequential and structural contexts. In the example above, the secondary structure assignment of the 7-th residue D is dependent on its local sequence context, and a sequence context window of size 7 is labelled in green. The local sequential contexts are represented in multiple ways, including sequential motifs that are manually curated, sequential similarities that are calculated based on homological profiles, and vector representations of neighbouring residues that are concatenated. The context2vec provides a vector representation for sequential patterns of local contexts. It is technically convenient to be combined with existing residue level representations that are generated from homological profiles, physicochemical properties and structural properties.

**Biological interaction-level predictions.** For predicting novel interactions in biological networks, for example, drug-target interactions, we proposed the prot2vec-based kernels as the input feature in kernel-based machine learning methods. This is based on the assumptions that if two proteins are similar to each other, they are likely to have interactions with the same third party. Let  $p_A$  and  $p_B$  denote the protein  $A$  and

protein  $B$ , respectively, and  $C$  the third party that has interaction with protein  $A$ . The similarity assumption can be represented as  $similar(p_A, p_B) \wedge interact(p_A, C) \rightarrow interact(p_B, C)$ . The third part  $C$  may refers to another protein  $p_C$  in protein-protein interaction prediction, and an interacting drug  $d_C$  in drug-target interaction prediction.

In kernel-based machine learning methods, protein similarity is calculated using different kernels, including the cosine kernel ( $k_{cos}$ ), the linear kernel ( $k_{lnr}$ ), the polynomial kernel ( $k_{pol}$ ), the RBF kernel ( $k_{rbf}$ ), the Laplacian kernels ( $k_{lap}$ ), and the sigmoid kernels ( $k_{sig}$ ). In this section, we introduce the prot2vec-based protein kernels, termed *vec2kernel*, by applying the aforementioned kernel functions to the generated prot2vecs of pairs of proteins.

Let  $v_{p_i}$  and  $v_{p_j}$  denote the prot2vecs generated for the  $i$ -th and  $j$ -th protein, respectively, the protein kernel  $K_{ij}$  can be formalised as follows,

$$K_{ij} = k_{\sigma}(v_{p_i}, v_{p_j}) \quad [\text{Eq. 4.5}]$$

where  $\sigma = cos, lnr, pol, rbf, lap, sig, \dots$

The purpose of biological interaction prediction is to determine whether there is a potential interaction between a protein and a third-party object, given the protein sequences of all proteins and the existing interactions status for part of the proteins. Let  $M$  denote the model that is constructed to predict biological interactions,  $p_i$  and  $p_j$  an arbitrary pair of proteins, and  $C$  the third-party object who has interaction with protein  $p_j$ . The prediction of biological interactions can be formalised as follows,

$$M: f(k_{\sigma}(v_{p_i}, v_{p_j}) | interact(p_j, C)) \rightarrow \begin{cases} 1 \dots if \text{ } interact(p_j, C) \\ 0 \dots if \text{ } !interact(p_j, C) \end{cases} \quad [\text{Eq. 4.6}]$$

where  $interact(a, b)$  and  $!interact(a, b)$  indicates that there is and there is not interaction between  $a$  and  $b$ , respectively. Similar to whole-sequence level predictions and residue level prediction, the key component here is to construct the model  $M$  that

maps the protein kernel between protein  $p_i$  and  $p_j$  to categorical values that indicate interaction status.

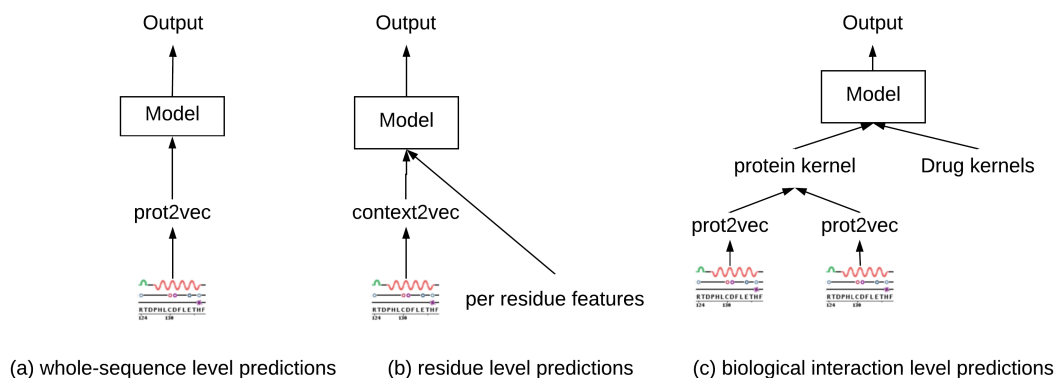
In this application scenario, the distributed representation of protein sequences,  $\text{prot2vec}$ , is used as the vector representation based on which multiple protein kernels can be conveniently calculated. Since  $\text{prot2vec}$  itself is generated to encode protein sequential patterns, it can be regarded as a machine interpretation of the protein sequence similarity-based kernel, which is directly generated from the sequential similarity between pairs of protein sequences. However, sequential similarities calculated with distance measurement scores are numeric values that cannot be transformed to other higher dimensional or lower dimensional spaces, whereas  $\text{prot2vecs}$  are vector representations that can be transformed to different kernel spaces using different kernel functions.

#### 4.1.5 Conclusion

In this section, we proposed three distributed representation of protein sequences,  $\text{prot2vec}^{\text{fetch}}$ ,  $\text{prot2vec}^{\text{non-overlap}}$  and  $\text{prot2vec}^{\text{inference}}$ , together with the previously proposed  $\text{prot2vec}^{\text{add}}$  [193], and introduced the construction methodology of each of them. We then summarised the advantages and disadvantages of the four vector representations. In order to validate the legitimacy of the training process and to tune the hyper-parameters of the two inference-based vector representations,  $\text{prot2vec}^{\text{non-overlap}}$  and  $\text{prot2vec}^{\text{inference}}$ , we further conducted comparison experiment to select the best performing generative models for respective representations. These two models will be used to generate  $\text{prot2vec}^{\text{non-overlap}}$  and  $\text{prot2vec}^{\text{inference}}$  in the rest of the thesis, for further application in protein sequence-based prediction tasks.

Based on the generated  $\text{prot2vecs}$ , we further proposed the application framework of the distributed representation of protein sequence in three different scenarios, *i.e.* whole sequence-level predictions, residue-level predictions, and biological interaction-level predictions, as is shown in **Figure 4.3**.





**Figure 4.3** The application framework of prot2vecs in three different scenarios.

For each of the application scenarios, we slightly modified the prot2vec by performing pre-processing and post-processing of its vector representation. More specifically, we directly used prot2vec as the feature vector for whole-sequence level prediction, proposed the context2vec as the contextual feature vector for residue level prediction, and applied multiple kernel functions based on prot2vec for kernel-based machine learning methods. The effectiveness of prot2vecs and its application framework is validated in real-world application tasks that are introduced in **Chapter 5, Section 7.4** and **Chapter 8**. For whole-sequence level predictions, we apply the prot2vec directly to protein family prediction. For residue level predictions, we apply the context2vec to the prediction of phosphorylation sites. For biological interaction prediction, we apply the kernels of prot2vecs to drug-target interaction prediction.

## 4.2 Deep learning framework for protein sequence-based predictions

In this section, we propose a deep convolutional neural network framework, namely *DeepS(equence)2P(roperty)*, for predicting protein structural and functional properties directly from protein sequences. This deep learning framework represents a template

framework that can be adjusted and applied to predict any residue-level protein structural and functional property.

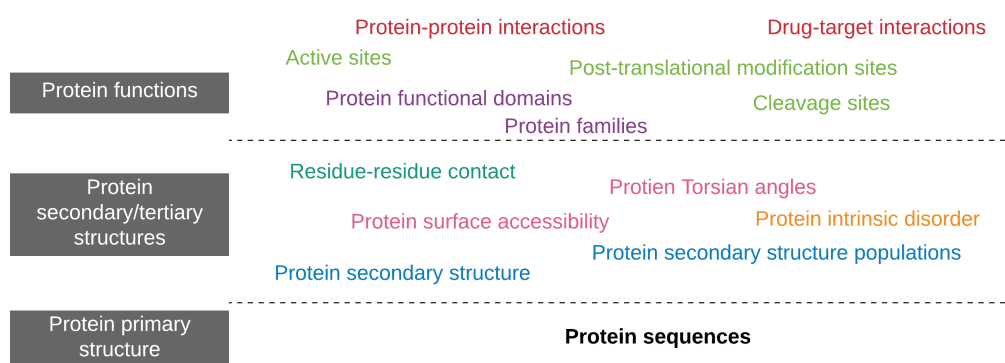
#### **4.2.1 Protein structural and functional properties**

Computational identification of protein structures represents a general research topic in bioinformatics. The first large-scale assessment of protein structure prediction methods was held in 1994 during the first meeting of the Critical Assessment of techniques for protein Structure Prediction (CASP1) [292]. Before 2016, twelve CASP assessments have been organised to collect protein targets, prediction methods and assessment methodologies for computational modelling of protein structures. The evaluated methods were used to model the prediction of protein structures including protein tertiary structures, protein secondary structures, fold recognition, structural domain boundaries, residue-residue contacts, disordered regions, protein functions. During the same time, other aspects of the protein structure were modelled beyond the CASP assessment, including protein torsion angles, backbone angles, dihedral angles, surface accessibilities, and side-chain conformations. In this thesis, we refer to the different aspects of protein structures as protein structural properties. Existing methods for predicting protein structural properties mainly rely on identified protein sequences, calculated protein homology profiles, derived protein physicochemical properties, and predicted related protein structural properties.

In the Swiss-Prot/UniProt database [39], protein sequences are annotated with seven categories of information, where four categories are directly related to protein functions. They are molecule processing, regions, sites, and amino acid modifications. These sequential annotations are position-specific, which means that each residue in protein sequences acquires an annotation label depending on the annotation type. For example, active sites in enzymes refer to the residues that are involved in the activity of protein enzymes, post-translational modification sites in amino acid modifications refer to the residues in protein substrates that are catalysed by protein kinases, while DNA binding

sites refer to the types and positions of the DNA binding domain. In this thesis, we refer to the above protein functional annotations as protein functional properties. Similar to the prediction of protein structural properties, the prediction of protein functional properties relies on protein sequences, calculated protein homology profiles, derived physicochemical properties, and predicted or determined protein structural properties.

Protein structural and functional properties are commonly predicted from protein sequences for three reasons. Firstly, the acquisition of protein sequences is easy and cheap. Secondly, protein sequences theoretically encode every information for determining the tertiary structures of proteins, even though the mechanism of protein folding is still unknown. Finally, many bioinformatics tools have been developed to derive a variety of protein information, such as homology profiles, physicochemical properties, and relevant protein structures, which provide sufficient sources of protein features to select from. The most widely used machine learning models include support vector machines, random forests, and different variations of neural networks. **Figure 4.4** demonstrates a subset of protein structural and functional properties.



**Figure 4.4** Protein sequence, protein structural properties and protein functional properties.

#### 4.2.2 Protein sequence-based residue-level predictions

In protein sequence-based residue-level predictions, each residue in protein sequences is assigned a specific structural or functional label annotation depending on the

prediction task. For example, in intrinsically disordered region prediction, each residue is assigned a binary class label indicating whether or not the residue is within a disordered region or not, while in 3-class protein secondary structure prediction, each residue is assigned one of the three class labels (Helix(H), Strands(E) and Coil(C)) indicating the secondary structure element that is assigned to the residue.

Therefore, the prediction task  $M_P$  for structural or functional property  $P$  can be modelled as follows,

$$M_P: f(r_i) \rightarrow c \quad [\text{Eq. 4.7}]$$

where  $i \in [1, \dots, |S|]$  indicates the position in the protein sequence of length  $|S|$ , and  $c$  represents the predicted structural or functional property for the target residue  $r_i$ , which can be categorical output in classification tasks or numeric output in regression tasks.

The above modelling process is only applicable for sequential annotation tasks which produce prediction for each position in the input of amino acid sequences. It is not applicable to residue-residue contact prediction, for example, which produces interaction/contact predictions between pairs of residues in protein sequences.

### 4.2.3 Feature selection, construction and extraction

In **Section 4.1**, we introduced the distributed representation, *i.e.* prot2vec, as a potential complementary feature representation for protein sequence-based predictions. Considering other protein sequence representations that are introduced in **Section 2.2.1**, there is a pool of information from which the most relevant and effective protein sequence representation can be selected, constructed and extracted.

Firstly, with *feature selection*, sources of information that are most relevant to the prediction task are selected, mainly rely on the understanding of biological background. The identification of phosphorylation sites, as an example, is considered to be related to the following factors [221],

- 1) Phosphorylation sites are enriched in specific secondary structures [293].

- 2) The identification of intrinsically disordered regions is helpful in improving the distinguish between phosphosites and non-phosphosites [217].
- 3) Phosphorylation sites are located at the surface of substrate protein sequences. Therefore, they may share similar patterns in surface accessible areas [222].
- 4) Phosphorylation sites need to be exposed to solvents [294], which is closely related to the average cumulative hydrophobicity.

Besides the sources of information that are experimentally proved to be correlated to the prediction task, features are also constructed from the sources of information that is used by state-of-the-art methods. For phosphorylation site prediction, there are dozens of existing machine learning-based methods among which the sequence similarities, homology profiles and physicochemical properties were also selected as the sources of information for calculating the input features. In general, the sources of protein information can be categorised into four types, including sequence similarity -based features, homology profile –based features, physicochemical property –based features, and structural features. **Table 4.4** summarises the most common features that are used in each of the four feature types.

**Table 4.4** A summary of the input features for the prediction of protein structural and functional properties.

Type name	Features
Sequence-based	1- 1-of-K coding 2- K-nearest neighbour (KNN) score 3- sequence alignment (similarity) search 4- sparse representation of protein sequences 5- distributed representations, etc.
Homology-based	1- Position-specific scoring matrix (PSSM) 2- HMM profile, etc.
Physicochemical property based	1- Overlapping properties 2- amino acid composition 3- polarity, hydrophobicity 4- surface tension 5- charge 6- normalised Van der Waals volume 7- polarizability 8- molecular weight 9- solubility 10- number of hydrogen bond donor in the side chain 11- number of hydrogen bond acceptor in the side chain, etc.
Structural based	1- Protein secondary structures 2- disordered proteins/regions 3- surface accessible area 4- solvent accessibility, etc.

Depending on the target structural and functional property of the prediction tasks, a subset of the features listed in **Table 4.4** should be used for the construction and extraction of input feature vector.

Secondly, *feature construction* refers to the process of constructing feature vectors from the selected sources of information. According to our empirical experience, for protein structural and functional prediction, this step represents the most time-consuming and complex procedure in the whole process of prediction task modelling. It involves data collection from a variety of relevant databases, familiarisation of publicly available source code packages and web servers, and the actual construction of feature vectors for respective selected features. For example, the calculation of homology profiles requires the acquisition of protein sequences from databases such as Swiss-Prot/UniProt [39] and RCSB PDB [40], the running of PSI-BLAST tool [99] for the calculation of *position-specific scoring matrixes (PSSMs)*, and potential post-processing to generate more informative numeric vectors such as Shannon entropy [221] and relative entropy [221]. Moreover, the calculation of some of the other features may rely on the results of PSSMs, such as protein secondary structures and intrinsically disordered states. The physicochemical property -based features, as another example, are usually encoded in various descriptors, such as the composition (C), transition (T), and distribution (D) descriptors, using different calculation formulae [295]. van Westen, G.J. et al. (2013) evaluated 13 different descriptors that were generated from a set of physicochemical properties [202]. In general, the calculated feature vectors from various selected sources of information are concatenated as one-dimensional feature vectors, which are later fed as the input for machine learning models.

Finally, *feature extraction* refers to the step of dimension reduction that is performed to generate more abstract vector representations with higher quality. The distributed representation of protein sequences introduced in **Section 4.1** was optimised for the task

of protein sequential modelling and can be regarded as the feature extraction from amino acid sequence -based sparse representations.

In this section, we propose the general convolutional neural network (CNN) for predicting protein structural and functional properties. For specific tasks, the vector representations constructed in accordance with the feature selection and feature construction are used as the direct input feature vector for the deep learning frameworks. During the process of training of the task-specific models that are derived from this general deep learning framework, the trained hidden layers perform the third step of feature extraction. For each protein sequence  $S$ , the three steps of feature selection, feature construction and feature extraction can be represented as follows,

$$f_{Sel} = select\_from (f_{sequence}, f_{homology}, f_{physicochemical}, f_{structural}) \quad [\text{Eq. 4.8}]$$

$$f_{Const} = concatenate(f_1, f_2, \dots, f_n) \dots f_i \in f_{Sel} \quad [\text{Eq. 4.9}]$$

$$f_{Extr} = h_d h_{d-1}, \dots, h_1(f_{construct}) \quad [\text{Eq. 4.10}]$$

where  $f_{Sel}$ ,  $f_{Const}$ ,  $f_{Extr}$  represent the selected, constructed and extracted features vectors respectively, and  $h_1, h_2, \dots, h_d$  represents the first, secondary and  $d$ -th hidden layers that are trained to optimally extract high abstract representations.

Technically, the hidden layers in the proposed CNN can be regarded as a feature extractor that can automatically extract high-level abstractive features from the raw data. Compared to human-engineered features, the extraction of these high-level abstractive features does not require human domain knowledge nor any human labour. The prediction performance of these abstractive features is validated and compared to human-engineered features in **Section 6.2.4**, **Section 6.3.5**, **Section 6.3.7**, and **Section 6.4.3** for predicting protein intrinsic disorder (IDP), protein secondary structure population (SSP) and IDP/SSP simultaneously.



#### 4.2.4 DeepS2P: a deep learning framework for protein sequence-based predictions

The general deep learning framework for protein sequence-based residue-level prediction is implemented as a convolutional neural network (CNN) [111] where each unit in the current hidden layer is connected to a restricted number of neighbouring units in the previous layer. The proposed deep learning framework for protein sequence-based predictions is designed for predicting structural or functional properties for each target residue  $r_i$  in each protein sequence  $S$ . It consists of multiple layers of computation units, including

**The input layer  $I$ .** A sliding window of size  $L$  is used to extract the neighbouring residues of the target residue  $r_i$ . Therefore, the feature vectors of the sequence segment  $s_{r_i}' = r_{i-w} \dots r_i \dots r_{i+w}$  of size  $L = 2 * w + 1$  are combined together to form an input feature vector  $v_I$  of size  $L \times |f_{Const}|$ , where  $f_{Const}$  refers to the constructed feature vector of each residue in  $s_{r_i}'$  according to the steps of feature selection and feature construction introduced in **Section 4.2.3**. With the input feature vector of size  $L \times |f_{Const}|$ , the input layer  $I$  is designed with  $L$  neurons, where each neuron has  $|f_{Const}|$  channels.

**The hidden layers  $H$ .** The hidden layers are used to extract high abstract vector representations that are optimised for the prediction of the target structural or functional property.

- Convolutional filter. In this design, we employ the architecture of convolutional neural networks, where each hidden layer is also called a convolutional filter. More specifically, each neuron in hidden layer  $h_i$  is only connected to a restricted number of neighbouring neurons in its previous hidden layer  $h_{i-1}$ . Let  $a_i^j$  denote the  $j$ -th neuron in hidden layer  $h_i$ . When the convolutional filter size  $f_{s_i}$  is set to 5, the input of the neuron  $a_i^j$  is computed from the output of the neurons  $a_{i-1}^{j-2}$ ,  $a_{i-1}^{j-1}$ ,  $a_{i-1}^j$ ,  $a_{i-1}^{j+1}$ , and  $a_{i-1}^{j+2}$ .

- Activation function. For each hidden layer, an *activation function* is added to add non-linearity to the modelling framework. Let  $o_i^j$  denote the linear weighted sum of the output of neuron  $a_{i-1}^{j-2}$ ,  $a_{i-1}^{j-1}$ ,  $a_{i-1}^j$ ,  $a_{i-1}^{j+1}$ , and  $a_{i-1}^{j+2}$ , and  $\sigma$  the activation function for the hidden layer  $h_i$ , the output of the neuron  $a_i^j$  is as follows,

$$\begin{aligned}
o_i^{j'} = \sigma(o_i^j) = & \sigma(w_{i-1}^{j-2} \times a_{i-1}^{j-2} + w_{i-1}^{j-1} \times a_{i-1}^{j-1} \\
& + w_{i-1}^j \times a_{i-1}^j + w_{i-1}^{j+1} \times a_{i-1}^{j+1} \\
& + w_{i-1}^{j+2} \times a_{i-1}^{j+2} + b_{i-1}^j)
\end{aligned} \tag{Eq. 4.11}$$

- Zero padding. The application of convolutional filter results in the decreased of the number of neurons in the next hidden layer. However, in protein sequence segments, each residue is equally important, and the information of the first and last several residues should be maintained in the convolved next hidden layer. Therefore, we applied the zero padding, where  $(fs_i - 1)/2$  ZERO neurons in layer  $h_{i-1}$  are added to ensure the calculated inputs for neuron  $a_i^0, a_i^1, \dots, a_i^{L-1}, a_i^L$  in hidden layer  $h_i$  (assuming the  $fs_i=5$ ).
- Max-pooling. Due to the lack of training data in most protein structural and functional property prediction tasks, over-fitting is more likely to be problematic for the training of the proposed deep learning framework. For the designing of the architecture of convolutional networks, the *max-pooling* technique is used to down-sample the outputs of each hidden layer. Let  $ps_i$  denote the filter size of the max-pooling layer that is added on the hidden layer  $h_i$  and  $L$  the number of neurons in  $h_i$ , the resulting number of neurons after the max-pooling layer is  $L/ps_i + 1$ .

The task-specific architecture for the hidden layer  $H$  relies on the experimental tuning of the following hyper-parameters, including the number of hidden layers, the convolutional filter size  $fs_i$ , and the max-pooling filter size  $ps_i$ .

**The fully-connected layers  $FC$ .** The *fully-connected layer* transforms the high-dimensional output of from the last hidden layer to a one-dimensional flat vector representation that produces the input for the output layers. Let  $F$  denote the number of neurons in the fully-connected layer, the resulting vector representation from the fully-connected layer is of the size  $F \times 1$ . The originally constructed input feature vector of the size  $L \times |f_{const}|$  is transformed to a high abstract and more condensed vector representation of size  $F \times 1$ , which can be regarded as the third step of feature extraction introduced in **Section 4.2.3**.

- **Drop-out.** In the architecture of DeepS2P, another technique, drop-out, was used to overcome the issue of over-fitting during the process of training. The drop-out layer is added on top of the last fully-connected layer and therefore is applied on the activations that are generated from the FC layer. Let  $\vartheta$  denote the percentage of activations from the last FC layer that is to be dropped during the process of training, only  $1 - \vartheta$  percentage of the activations will be considered as the input for the output layer.

The task-specific architecture for the fully-connected layer and drop-out layer relies on the experimental tuning of the following hyper-parameters, including the number of fully connected layers, and the number of neurons  $F$  in each FC layer, and the drop-out percentage  $\vartheta$ .

**The output layer  $O$ .** Depending on specific prediction tasks, this layer uses  $N$  real-valued neurons representing  $N$  number of classes to be predicted. For classification tasks, the *sigmoid function* [Eq. 4.12] was used as the activation function and *the mean squared error (MSE)* [Eq. 4.14] as the loss function. For regression tasks, the *cross*

entropy (CE) [Eq. 4.15] was used as the loss function and the softmax function [Eq. 4.13] as the activation function.

$$\text{sigmoid}(x) = \frac{1}{1+e^{-x}} = \frac{e^x}{e^x+1} \quad [\text{Eq. 4.12}]$$

$$\text{softmax}(x)_j = \frac{e^{x_j}}{\sum_{k=1}^K e^{x_k}} \text{ for } j = 1 \dots K \quad [\text{Eq. 4.13}]$$

$$\text{MSE}(\hat{y}, y) = \frac{1}{n} \sum_{i=1}^n (\hat{y}_i - y_i)^2 \quad [\text{Eq. 4.14}]$$

$$\text{CE}(\hat{y}, y) = -\sum_{i=1}^n y_i * \log(\hat{y}_i) \quad [\text{Eq. 4.15}]$$

In equation [Eq. 4.12] and [Eq. 4.13],  $x$  is the weighted sum of the outputs from the drop-out layer. In equation [Eq. 4.14] and [Eq. 4.15],  $\hat{y}$  is the predicted results of the true label  $y$ .

#### 4.2.5 Hyper-parameters for DeepS2P

When applying the DeepS2P framework to predict different protein structural and functional properties, *hyper-parameters* determine the final structure of the task-specific models and the specification of training procedures, while parameters are automatically learned during the process of model training.

As partially introduced in **Section 4.2.4**, the hyper-parameters for determining the task-specific model structure include the number of neurons in the input layer  $L$ , the number of hidden layers  $|H|$ , the convolutional filter size  $f_{s_i}$  for each hidden layer, the filter size  $p_{s_i}$  for each max-pooling layer, the number of fully-connected layers  $|FC|$ , and the number of neurons in each fully-connected layer  $F_j$ .

At the same time, the process of model training is based on the specification of several training-related hyper-parameters including the number of training epochs  $m$ , the size of mini-batches  $n$ , the learning rate  $\alpha$ , the learning rate decay  $\delta$ , and the drop-

out rate  $\vartheta$ . **Table 4.5** demonstrates the hyper-parameters in DeepS2P and their corresponding functions in nailing the final structure of each task-specific model.

**Table 4.5** A summary of the hyper-parameters for the DeepS2P framework.

Name	Symbol	Description
<i>A. Structure determinant hyper-parameters</i>		
Size of the input layer	$L$	The size of the sliding window for extracting neighbouring residues of the target residue.
Number of hidden layers	$ H $	The number of hidden layers
Convolutional filter size	$fs_i$	The convolutional filter size for hidden layer $i$ .
Max-pooling filter size	$ps_i$	The max-pooling filter size for hidden layer $i$ .
Number of FC layers	$ FC $	The number of fully-connected layers.
Size of FC layers	$F_j$	The number of neurons in each fully-connected layer $j$ .
<i>B. Training process determinant hyper-parameters</i>		
Number of epochs	$m$	The number of times that the training data set is traversed through during the training process.
Size of mini-batches	$n$	The number of training examples in each mini-batch. Model parameters are updated for each mini-batch instead of each training example.
Learning rate	$\partial$	The scale of parameter updating for each mini-batch
Learning rate decay	$\delta$	The rate of decreasing the learning rate during the process of training.
Drop-out rate	$\vartheta$	The rate of drop-out activations during the process of training.

#### 4.2.6 Conclusion

In this section, we introduced a deep learning framework, DeepS2P, for protein structural and functional prediction based on protein sequences. We formalised protein sequence-based residue-level predictions, illustrated the process of feature selection, construction and extraction, and introduced the full architecture of the CNN-based framework. Different from existing applications of deep learning in protein analysis, which are conducted in an *ad hoc* manner, the architecture of DeepS2P is constructed as template framework so that it can be applied to a group of protein sequence-based prediction tasks. It allows the flexibility of the selection of input features, the design of the structure of the hidden layers (such as the number of hidden layers, the number of hidden units, and the use of max-pooling and dropout techniques), and the modelling of the expected prediction outputs. We also introduced the hyper-parameters that are used to determine the final structure of task-specific models and the hyper-parameters that are used to guide the actual training process of the constructed deep learning models. The DeepS2P framework provides the basic architecture of the multitask deep learning framework and the deep transfer deep learning framework that are introduced in **Section 4.3** and **Section 4.4**.

### 4.3 Multitask deep learning for protein structural prediction

In this section, we propose the application of two multitask deep learning framework, *i.e.* the *Multi-task* framework and the *Cross-stitch* framework, to predict two or more protein structural properties simultaneously. These two deep learning frameworks represent templates that can be adjusted and applied to predict any two or more related structural properties.

#### 4.3.1 Qualitative correlation between protein structural properties

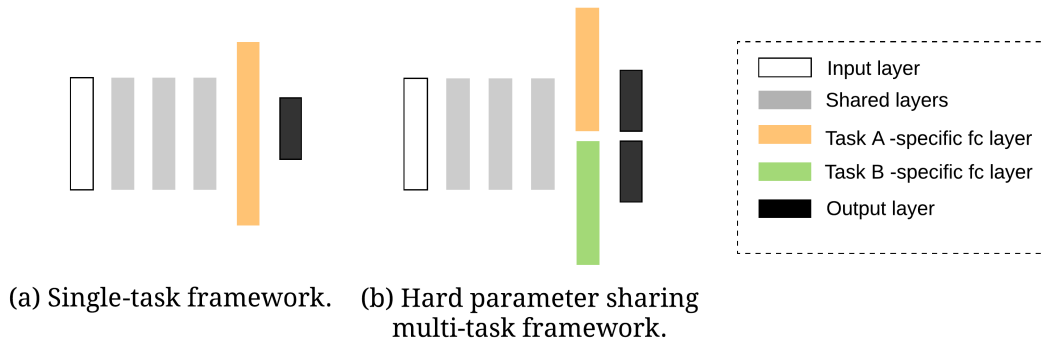
Protein structural properties are not independent from each other. In existing machine learning-based methods that are introduced in **Section 2.3**, the prediction of one protein structural property relies on the output of the predicted outputs of another protein structural property. For example, SPINE-D [212], DeepCNF-D [158], AUCpreD [207], and SPOT-disorder [153] all used the predicted secondary structure and accessible surface area as part of the input feature for predicting protein intrinsic disorder. The rationale behind these methods is that the residues annotated with the secondary structure element coil is more likely to be disordered [296]. Also, the prediction protein secondary structures and accessible surface areas are usually presented together (but are not necessarily predicted simultaneously) [8, 138]. According to Lins, L., Thomas, A. and Brasseur, R. (2003), the most accessible residues belong to the random coil/turn (C-T) class, whereas the *Ba* and *Bp* structures result in the most solvent-inaccessible residues [297]. According to these observations, there exist qualitative correlations among different protein structural properties.

In principle, better prediction performance of the prediction of individual protein structural properties can be obtained by predicting two or more related protein structural properties simultaneously within a multitask framework. More specifically, because of the qualitative correlation between protein structural properties, the representation learned for one task is expected to be partially useful for the prediction of its related prediction tasks, and vice versa. Therefore, in this section, we propose the application of multitask deep learning frameworks to the prediction of two or more protein structural properties simultaneously. In particular, we are seeking to explore the quantitative correlations between the simultaneously prediction protein structural properties by automatically learn the supportive weights between each two related prediction tasks.



### 4.3.2 Multitask deep learning with hard parameter sharing

As introduced in the **Section 2.1.4**, the deep learning implementation of multitask learning can be categorised into two types, *i.e.* the hard parameter sharing and the soft parameter sharing. With hard parameter sharing, it is straightforward to extend the DeepS2P framework for multitask learning. **Figure 4.5** demonstrates the architecture for the Multi-task framework based on hard parameter sharing.



**Figure 4.5** The difference between the architecture of DeepS2P and multitask framework with hard-parameter sharing.

**Figure 4.5** (a) demonstrates the simplified illustration of the deep learning framework DeepS2P, where the input layer, hidden layers, the fully-connected layer and output layer are denoted as white, grey, coloured and black squares respectively. In contrast, **Figure 4.5** (b) demonstrates the simplified illustration of the multitask framework via hard parameter sharing. With the hard parameter sharing, in terms of the framework architecture, the input layer and hidden layers are kept unchanged and shared among two or more prediction tasks. However, each prediction task has its task-specific fully-connected layer and output layer, for which the respective weights are not shared among tasks. The basic idea of this particular design is to allow the sharing of weights in hidden layers which are updated according to the training examples of all related tasks, but also allow a certain level of variations for individual tasks by themselves. In this thesis, we refer to this architecture of multitask framework as the Multi-task framework.

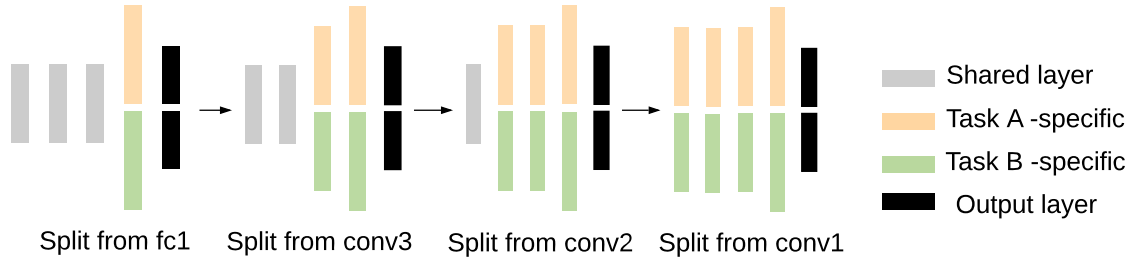
In the Multi-task framework, the set of shared hidden layers is actually the feature extractor that is optimized to perform all tasks. It has the advantage of decreasing the risk of over-fitting because 1) it regularizes the effect of each task on the process of weight updating and 2) it allows the training of the hidden layers for tasks that has limited training data. The disadvantage of this architecture is that it pre-defines the parameter sharing schema by specifying that weights in hidden layers have to be shared and weights in fully-connected layers have to be task-specific. In real-world application, we usually do not know the quantitative correlation between multiple protein structural properties in advance. Therefore, it is interesting to relax this parameter-sharing schema so that the dependence between multiple prediction tasks are explored automatically.

#### **4.3.3 Cross-stitch multitask learning for quantitative correlation analysis**

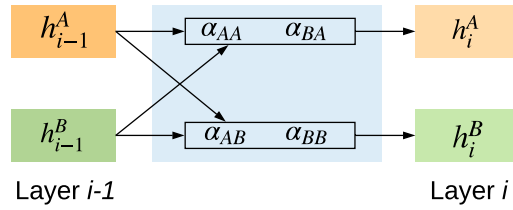
To automatically learn the quantitative correlation between related prediction tasks, we also designed a Cross-stitch multitask framework [182]. It is based on soft parameter sharing where both tasks have their own neural network models with multiple hidden layers. For each pair of corresponding hidden layers from the two tasks, a cross-stitch unit is added to linearly combine the outputs of the two hidden layers, producing inputs for their next hidden layers respectively. **Figure 4.6** (a) illustrates the architecture of the cross-stitch multitask deep learning framework for prediction two or more protein structural properties. It is based on the idea of soft parameter sharing, where each individual task has its own input layers, hidden layers, fully-connected layers and output layers.

Between each two individual tasks, a cross-stitch unit was added on top of each hidden layer and each fully-connected layer. They linearly connect the output from the previous layers and accordingly generate the input for the next layers respectively for the two tasks. The parameters of each of the cross-stitch units are automatically learned during the process of model training. Therefore, the parameter sharing between two tasks are not enforced. Instead, the dependencies between the models of the two tasks

are dynamically adjusted according to the correlation between these two tasks. With the cross-stitch multitask framework, it is possible, for the first time, to automatically learn the quantitative correlation between two tasks. This feature is especially interesting for the quantitative analysis of the correlation between two or more protein structural properties. In **Section 6.4.4**, we use a heat map to demonstrate the quantitative correlation between the predictions of IDP/IDR and SSP.



(a) Shared and split layers in deep convolutional neural networks



(b) Cross-stitch unit connecting neighboring layers in deep convolutional neural networks.

**Figure 4.6** The framework for a cross-stitch multitask deep convolutional neural network.

The cross-stitch unit between task A and task B for layer  $h_i$  in **Figure 4.6** (b) is composed of a matrix:

$$M = \begin{bmatrix} \alpha_{AA} & \alpha_{BA} \\ \alpha_{AB} & \alpha_{BB} \end{bmatrix} \quad [\text{Eq. 4.16}]$$

which indicates the linear relation between the contribution of the outputs from the

previous layers  $h_{i-1}^A$  and  $h_{i-1}^B$ . Let  $I_{h_i} = \begin{bmatrix} i_{h_i}^A \\ i_{h_i}^B \end{bmatrix}$  denote the inputs of layer  $h_i$ , where  $i_{h_i}^A$  and

$i_{h_i}^B$  denote the input of layer  $h_i$  for task A and task B, respectively; and  $O_{h_{i-1}} = \begin{bmatrix} o_{h_{i-1}}^A \\ o_{h_{i-1}}^B \end{bmatrix}$

denote the outputs of layer  $h_{i-1}$ , where  $o_{h_{i-1}}^A$  and  $o_{h_{i-1}}^B$  denote the output of layer  $h_{i-1}$  for task A and task B, respectively, the  $I_{h_i}$  in cross-stitch framework can be represented as follow,

$$I_{h_i} = MO_{h_{i-1}} \quad [\text{Eq. 4.17}]$$

As a result, the inputs for layer  $h_i$  for task A and task B can be respectively represented as:

$$i_{h_i}^A = \alpha_{AA} * o_{h_{i-1}}^A + \alpha_{BA} * o_{h_{i-1}}^B \quad [\text{Eq. 4.18}]$$

$$i_{h_i}^B = \alpha_{AB} * o_{h_{i-1}}^A + \alpha_{BB} * o_{h_{i-1}}^B \quad [\text{Eq. 4.19}]$$

We added a cross-stitch unit after each hidden layer and the fully-connected layer, yielding four cross-stitch units altogether.

#### 4.3.4 Alternate training

Despite the large amount of protein sequences that became available in the last 20 years, there is a lack of labelled data for protein structural and functional properties. Therefore, it is even more difficult to obtain a training dataset with annotations of multiple protein structural properties. In this thesis, we use the strategy of *alternate training* to address this issue. More specifically, we trained the model with mini-batches acquired from individual dataset for each protein structural property in an alternate manner. For each mini-batch, only one loss function from [Eq. 4.14] or [Eq. 4.15] was used to optimize the model. As a result, the shared parameters in hidden layers were updated in each mini-batch, while the task-specific weights were only updated in alternate mini-batches from respective tasks. Because there are effectively multiple loss functions in alternate training, the Multi-task framework can be optimized to favour one particular task with the supporting of the other task. Let A and B denote the two protein structural properties that are predicted simultaneously, to train a model favouring the prediction of A (Multi-task-A), we alternately train the model with data of A and B respectively, but only save the

model when the performance of the prediction of A was improved on the validation set of A.

#### 4.3.5 Conclusion

In this section, we introduced the multitask frameworks, Multi-task and Cross-stitch, for predicting two or more protein structural properties simultaneously. We demonstrated the rationale behind this particular design and introduced two implementations of the multitask learning based on the DeepS2P framework. We also discussed the challenge of training the models of these two frameworks when there is a lack of training data. The application of multitask deep learning framework to simultaneous prediction of protein structural properties demonstrates the importance of incorporating experimental and computational observations in the design of deep learning frameworks for protein analysis.

### 4.4 Deep transfer learning for functionally important site prediction

In this section, we propose the deep transfer learning framework, *DeepTransfer*, for predicting functionally important sites (FISs) (see **Section 1.2.3** for detailed introduction of FISs). According to **Appendix A**, functionally important sites can be categorised into six functional categories, and we particularly focus on the prediction of post-translational modification (PTM) sites whose catalytic enzymes are organised in hierarchical structures.

#### 4.4.1 Post-translational modification site prediction

As introduced in **Section 1.2.1**, post-translational modification (PTM) refers to the covalent and enzymatic modifications after translation [33]. It is unusually catalysed by a group of protein enzymes and causes modifications to specific residues in substrate proteins. These residues in substrate proteins that are catalysed during PTM are referred

to as *Post-translational modification sites (PTM sites)* and are considered functionally important. PTM sites are referred to as different sites depending on the type of the modification event. For example, the PTM phosphorylation is catalysed by *protein kinases* and the residues in substrate proteins that are modified are referred to as *phosphorylation sites*, while cleavage is catalysed by different *proteases* and the modified residues are referred as *cleavage sites*. Please refer to **Appendix A** for detailed information of PTM sites and their catalytic enzymes.

To better characterise the PTM events, various computational methods have been developed to predict PTM sites directly from protein sequences. Since enzymes function with high specificity, enzyme-specific predictive models which predicts PTM sites for specific enzymes are considered to have superior prediction performance. For example, in GPS 3.0 kinase-specific predictive models were constructed for 464 protein kinases<sup>22</sup> [77, 214, 215], while in PROSPER protease-specific predictive models were constructed for 24 protease families<sup>23</sup> [298].

Despite the progress made in these particular tasks, it remains a challenge to build enzyme-specific predictive models due to the lack of annotated sites for each enzyme. In phosphorylation site prediction, as an example, the protein kinase *PKC* had the most sufficient annotations of 962 phosphorylation sites while most protein kinases have hundreds or dozens of annotated phosphorylation sites, which is far less than the amount of training data that is required to train a deep learning model.

#### **4.4.2 Hierarchical data structures of proteins and enzymes**

Hierarchical classification systems are ubiquitous in biology. For example, the Gene Ontology [249] is defined as a loosely organized hierarchical structure where the child node represents a more specialized term than its parent node. In terms of proteins, a

---

<sup>22</sup> GPS 3.0: <http://gps.biocuckoo.org>

<sup>23</sup> PROSPER: <https://prosper.erc.monash.edu.au/home.html>

broadly applied classification system is used to organize proteins into *superfamilies*, *families* and *subfamilies* [70, 299], where proteins are assigned as leaf nodes in the hierarchical classification tree. For protein enzymes, the *Enzyme Commission number* (*E.C. number*) [250] are used to organise enzymes into a five-level (at most) hierarchical structure. And for protein kinase, a group of enzymes that catalyses phosphorylation, a four-level hierarchical classification tree is constructed according to the homology distance [300]. In these hierarchical classification trees, proteins with shorter tree path are considered to be more evolutionarily close to each other and therefore more likely to catalyse sites in substrates with similar local patterns.

#### 4.4.3 Hidden layers as feature extractors

The task of PTM site prediction can be modelled as the protein sequence-based residue-level binary classification problem which produces *PTM site* or *non-PTM site* for each target residue. Therefore, it is straightforward to apply the DeepS2P framework to predict PTM sites of specific enzymes. For a potential site  $s$ , the deep learning framework seeks to learn a model  $M_k$  of the specific enzyme  $k$  that can best predict the true label  $o$  indicating whether  $s$  is catalysed by  $k$  or not.

The DeepS2P framework can be regarded as the combination of a *representation extractor* (the hidden layers) that extracts highly abstract feature vectors from the local sequence segment of target residue  $r_i$  and a *binary classifier* (the fully connected layer and the output layer) that performs classification based on the extracted feature vectors. The advantage of the convolutional feature extractor is that multiple features of neighbouring residues of  $r_i$  are combined and represented as one highly abstract feature vector. This feature vector is automatically learned to specifically and optimally represent the local patterns for phosphorylation site prediction.

The disadvantage of the deep CNN is that it incorporates more parameters and therefore requires more training data for each enzyme. For enzymes with limited annotated sites, there is a tendency of overfitting during the process of training the deep

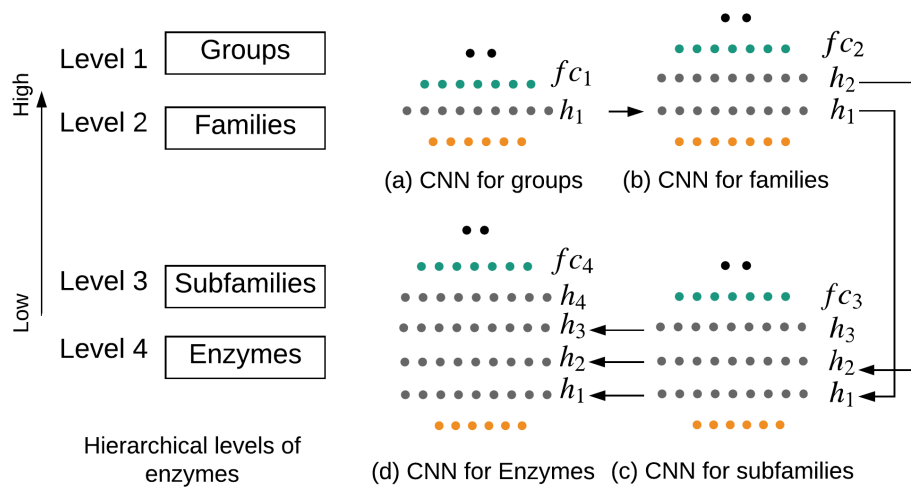
CNN. This is because there are not enough positive and negative examples for model generalization.

#### 4.4.4 Deep transfer learning for hierarchical enzymes

For enzyme-specific PTM site prediction, we mapped the hidden layer  $i$  in the generic deep CNN to level  $t$  in the *hierarchical enzyme classification system (HECT)*. Let  $x$  be the input of the deep CNN,  $h_i$  the convolutional filter at the  $i$ -th hidden layer, and  $y_i$  the binary output of functional site prediction. For enzyme nodes at the  $t$ -th level of the HECT,

$$y_t = \sigma(W_{to}h_t^{\theta_t}(h_{t-1}(\dots h_1(x)))) + b_{to}; \theta_t \quad [\text{Eq. 4.20}]$$

where  $h_1 \dots h_{t-1}$  are convolutional filters that are pre-trained by annotated sites of enzyme nodes at level 1 ...  $t - 1$ , respectively,  $h_t^{\theta_t}$  is the convolutional filter for enzyme nodes at level  $t$  whose parameters  $\theta_t$  are to be learned,  $\sigma$  is the activation function at the output layer, and  $W_{to}$  and  $b_{to}$  are the weight and bias parameter for the fully connected layer, respectively. The parameter  $\theta_t$  is only trainable for convolutional filter  $h_t$ , whose input feature is generated by applying a sequence of pre-trained convolutional filters  $h_1 \dots h_{t-1}$ .



**Figure 4.7** Deep transfer learning for kinase-specific phosphorylation site prediction.



**Figure 4.7** illustrates the deep transfer learning framework for PTM site prediction in accordance with the hierarchy of the catalytic enzymes.

#### 4.4.5 Overfitting and underfitting

Overfitting and underfitting are two opposite issues during the process of model training. The former is usually caused by the lack of training examples, the absence of regularisation, or a learning rate that is set too large, while the latter happens when the model cannot capture the underlying structure of the data.

For PTM site prediction in **Figure 4.7**, models of subfamilies and enzymes at the Level 3 and Level 4 are more likely to be trained with overfitting due to the lack of training examples. This issue is expected to be partially addressed by the level-by-level representation extraction using the deep transfer learning framework. However, in order to optimize the prediction performance for each enzyme node, we also employed the strategy of selecting the best performing model from the models of their ancestor nodes and themselves at Level 1, 2, 3 and 4 as the final model.

This post-processing step of model selection can be regarded as the searching for the balance between overfitting and underfitting. For enzyme  $k$ , the model trained for its subfamily  $s$  is theoretically under fitted (since the training examples of subfamily  $s$  contain annotated sites for other enzymes who belong to the same subfamily). Yet, it may have a better prediction performance compared to an overfitting model  $k$ , considering the structural similarities between sibling enzyme nodes that share the same parent in the hierarchical tree.

#### 4.4.6 Conclusion

In this section, we introduced the deep transfer learning framework for predicting functionally important sites of enzymes that are organized in hierarchical structure. Based on the observation of the ubiquitous hierarchical classification system in protein

data, we designed the transfer learning framework whereby each hidden layer corresponds to one level in the hierarchical classification system. This particular design is advantageous to prediction tasks where there is a lack of training data for leave nodes. It demonstrates the importance of incorporating biological domain knowledge to the design of the deep learning frameworks for protein analysis.

## 4.5 Chapter summary

In this chapter, we proposed four deep learning frameworks for the modelling of protein sequence, protein structures and protein functions.

The distributed representation of *prot2vec* is presented as a complementary representation of protein sequences. Compared to existing protein sequence representations that are generated with biological heuristics, the generation of the distributed representation of protein sequences is purely driven by large quantities of protein sequence data. We also introduced three application scenarios in which the distributed representations are used for whole sequence-level, residue-level and biological interaction-level prediction tasks.

We then introduced the deep learning framework, *DeepS2P*, that can be applied to predict any residue-level protein structural and functional properties. This framework represents the basic architecture that is later extended for specific applications. The application of *DeepS2P* in practice requires the selection and construction of input features, the determination of hyper-parameters and the process of model training.

Based on the *DeepS2P* framework, we proposed the multitask deep learning framework *Multi-task* and *Cross-stitch* based on the qualitative correlations among protein structural properties and the deep transfer learning framework *DeepTransfer* based on the observation of the ubiquitous hierarchical classification systems. The multitask learning frameworks allow for the simultaneous prediction of two or more protein structural properties and automatically explores the quantitative correlation between two or more protein structural properties. The deep transfer learning framework

provides a transferrable way of sharing pre-trained models for enzymes that are organized in hierarchical structures. **Table 4.6** summarises the deep learning frameworks that were proposed in this chapter and their application scenarios.

**Table 4.6** A summary of deep learning frameworks and their application scenarios.

Frameworks	Application scenarios
prot2vec	Whole sequence-level predictions; Residue-level predictions; Biological interaction-level predictions.
DeepS2P	Protein sequence-based residue-level predictions.
Multi-task	Simultaneous predictions of two or more protein structural properties.
Cross-stitch	Simultaneous predictions of two or more protein structural properties, and automatic discovery of the quantitative relation between two or more protein structural properties.
DeepTransfer	Functionally important site prediction for enzymes in hierarchical classification systems.

In **Chapter 5 – Chapter 8**, we introduce the applications of the proposed deep learning frameworks to specific prediction tasks. In protein family prediction, we demonstrate the performance of prot2vecs as the direct input feature for whole-sequence level prediction tasks. In phosphorylation site prediction, we validate the performance of context2vec as a complementary input feature for residue-level prediction tasks. We also applied the DeepS2P and DeepTransfer framework to predict kinase-specific phosphorylation site prediction with limited training data. In the prediction of protein intrinsic disorder and secondary structure populations, we applied the DeepS2P framework and the multitask deep learning frameworks to demonstrate the positive effect of predicting multiple protein structural properties simultaneously. Finally, in drug-target interaction prediction, we evaluate the performance of the vec2kernel representation for biological interaction-level prediction tasks.

# 5 Protein family prediction

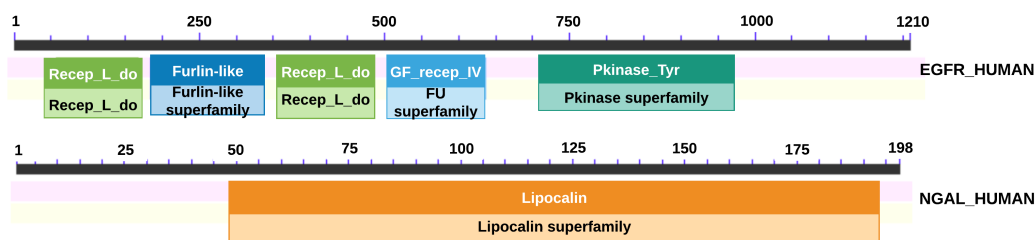
## 5.1 Introduction

In **Section 4.1.4**, we proposed the application framework of the distributed representation of protein sequences, *i.e.* *prot2vec*, in three different scenarios. In this chapter, we introduce the application of *prot2vec* in protein family prediction to validate the effectiveness of *prot2vec* as the direct input feature in whole sequence-level prediction tasks. The method for predicting protein family based on *prot2vec* is referred to as *PfamProt2vec*.

The constructed benchmark for the independent test and the web service for *prot2vec*-based protein family classification system *PfamProt2vec* are publicly available at <http://118.138.240.130>.

## 5.2 An overview of protein family prediction

Protein families refer to groups of proteins that share a common evolutionary origin, reflected by their related functions and similarities in their sequences and structures [301]. In early years, during which time only small and *single-functional* proteins such as lipocalin were structurally understood [302], protein family referred to the cluster of proteins that are evolutionally related and thus have similar functions. However, more proteins were found to be *multiple-functional* [303], where multiple functional domains (see **Section 1.2.3**) are evolved independently over time within the same protein sequence, *i.e.* *domain shuffling* [304]. Since then, family-based and domain-based classification of proteins is usually addressed simultaneously. **Figure 5.1** demonstrates the multiple functional domains in protein EGFR\_HUMAN and the single functional protein NGAL\_HUMAN. These results were returned from searches performed against the database Pfam 30.0 [70] with the expected value threshold set to 0.01.



**Figure 5.1** The searched protein domain for protein EGFR\_HUMAN and NGAL\_HUMAN returned by CDD.

In **Figure 5.1**, four Pfam domain hits (protein family hits) were returned for protein EGFR\_HUMAN, including *the Protein tyrosine kinase* (PF07714), *the Growth factor receptor domain IV* (PF14843), *the Furin-like cysteine-rich region* (PF00757), and *the Receptor L domain* (PF01030). Even though each of the four returned Pfam domains in EGFR\_HUMAN only occupies a short segment of the protein sequence, the protein EGFR\_HUMAN can be classified as a family member of the protein family PF07714, PF14843, PF00757 and PF01030, simultaneously. However, for protein NGAL\_HUMAN, only one Pfam family was returned, *i.e. the Lipocalin family* (PF00061) which occupies 75% of the sequence. Proteins like NGAL\_HUMAN have only one domain, therefore, can only be classified to one protein family.

To functionally annotate proteins at a large scale, several computational classification systems are constructed by clustering proteins into protein families [70, 301]. In this thesis, this particular task of classifying proteins to construct protein classification system from scratch is referred to as *protein family classification*. According to the information based on which proteins are classified, these classification systems can be categorised into four types including a) sequence-based systems such as PROSITE [68], b) homology-based systems such as Pfam [70], c) structure-based

systems such as CATH [88], and d) meta-classification systems<sup>24</sup> such as InterPro [85, 86]. However, there is not such a standard classification system that is perfectly accurate and different classification systems provide different interpretations of protein families based on protein sequential similarities, structural similarities and functional relations.

In this thesis, we refer to the task of training models to classify proteins into an existing protein family classification system as *protein family prediction*. To quantitatively evaluate prot2vec as an effective feature vector for protein family prediction, we train predictive models based on one of the existing constructed protein family classification systems. It is different from protein family classification in that it assumes an existing classification system to be the standard classification system, based on which the performance of prot2vec can be evaluated.

For prot2vec-based protein family prediction, we use prot2vec as the vector representation of protein sequences, the SVM [86, 145] as the classification algorithm, and the Pfam database [70] and the InterPro database [85] as the standard classification systems based on which we validate the effectiveness of prot2vec.

### 5.3 Dataset construction

In this chapter, we conduct a series of quantitative investigations to systematically analyse the effectiveness of prot2vec as an input feature vector for protein family prediction. For these investigations we constructed two overall datasets from the Pfam database [70] and the InterPro database [85, 86], resulting in *the UniProt-Pfam dataset* and *the UniProt-InterPro dataset*, respectively.

---

<sup>24</sup> Meta classification systems combines the database of multiple classification systems and provide a consistent interface to database users.

### 5.3.1 The UniProt-Pfam dataset

The UniProt-Pfam dataset is constructed using the following procedure. Firstly, we downloaded the raw data from the Swiss-Prot/UniProt database (Sep 2016), containing 554,171 reviewed proteins. Secondly, we extracted data lines starting with 'AC' and 'DR Pfam'<sup>25</sup>, resulting in pairs of proteins and families  $(p, f)$  indicating that protein  $p$  is a member of the protein family  $f$ . Thirdly, we transformed the pairs of protein and their families into sets of proteins, where each set contains protein members that belong to the same protein family. Finally, we filtered out proteins whose protein sequences are not identified yet. This procedure left us with 9,337 protein families with their respective protein members, covering a total of 366,776 proteins.

Due to the imbalanced sizes of proteins that belong and do not belong to a specific protein family, we conducted negative sampling for each protein family to balance out the number of positive and negative examples. More specifically, we randomly selected an equal number of negative examples and positive examples, resulting in a training dataset whose positive to negative example ratio is 1:1, for each extracted protein family.

### 5.3.2 The UniProt-InterPro dataset

The UniProt-InterPro dataset is constructed using the following procedure. We extracted from the InterPro database [85] (downloaded in July 2017) 551,047 proteins in 26,239 InterPro protein families/domains, containing 4,408,698 protein family annotations. We then filtered out protein families that have less than 100 protein members, resulting in 426,414 reviewed proteins in 1,461 protein families. Similar to the UniProt-Pfam dataset, the negative sampling was performed to ensure a ratio of 1:1 between positive and negative examples.

---

<sup>25</sup> In the Swiss-Prot/UniProt database, the protein family annotations are recorded in the 'AC' segment with the 'DR Pfam' label.

## 5.4 Vector space analysis for human protein families

To validate the effectiveness of prot2vec as a feature vector for protein family prediction, we performed the prot2vec-based vector space analysis with respect to different protein families. We found that protein families that were closer to each other in the prot2vec-based vector space possessed a higher functional correlation. In contrast, protein families that belong to different functional categories were more separated in the vector space. In this investigation, the implementation prot2vec<sup>inference</sup> (see **Figure 4.2**) was used to generate the distributed representation of protein sequences.

### 5.4.1 Experimental design

From the constructed UniProt-Pfam dataset with 366,776 proteins, we selected 17,508 Human proteins across 5,736 protein families. We then removed protein families that had less than 100 proteins and ended up with 25 protein families for follow-up analysis.

We first ranked each pair of protein families (PFs) according to their prot2vec-based cosine distance described below and then examined the relationship between the two protein families in top-ranked PF pairs, in terms of the largest-to-smallest ranking and the smallest-to-largest ranking, respectively. To calculate the prot2vec-based distance between the protein family A and the protein family B, we propose *the average normalised cosine distance* which is defined as follows,

$$Distance(A, B) = \frac{Dist_{AVG}(\hat{A}, \hat{B})}{Dist_{AVG}(\hat{A}, \hat{A}) + Dist_{AVG}(\hat{B}, \hat{B})} \quad [\text{Eq. 5.1}]$$

$$\text{where } Dist_{AVG}(\hat{A}, \hat{B}) = \frac{\sum_{i=1}^{N_{\hat{A}}} \sum_{j=1}^{N_{\hat{B}}} \cos_{distance}(v_{\hat{A}_i}, v_{\hat{B}_j})}{N_{\hat{A}} \times N_{\hat{B}}}$$

The distance  $Distance(A, B)$  between protein family A and B is calculated as the average cosine distance between two sets of protein members  $\hat{A}$  and  $\hat{B}$ . Here,  $N_{\hat{A}}$  and



$N_{\hat{B}}$  denote the number of protein members in A and B, respectively, while  $v_{\hat{A}_i}$  and  $v_{\hat{B}_j}$  denote the prot2vec of the  $i$ -th and  $j$ -th member in A and B, respectively. The normalisation term  $Dist_{AVG}(\hat{A}, \hat{A}) + Dist_{AVG}(\hat{B}, \hat{B})$  is employed as a penalisation because some PFs are more broadly defined and thus have a larger average distance among their own protein members. Based on the calculation, we ranked the 125 (=25×25) pairs of PFs in terms of their normalised prot2vec-based cosine distances.

#### 5.4.2 Results analysis

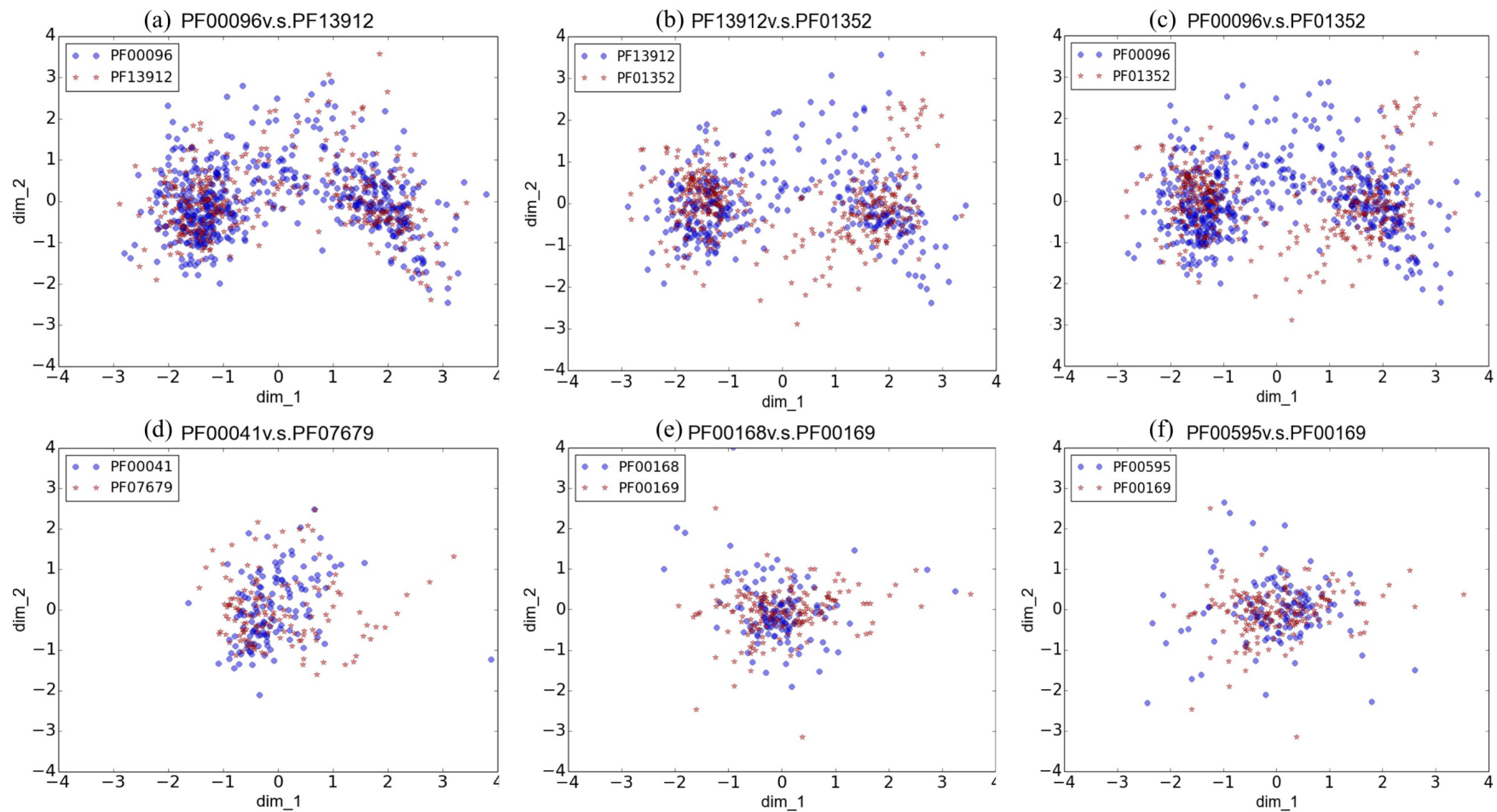
**Table 5.1** listed the top six PF pairs whose members were closest to each other, as well as the top six PF pairs whose members were most distant from each other, according to the calculated normalised prot2vec-based cosine distances. For each of the listed PF pairs, we visualised their member proteins in the 2-dimensional vector space, by mapping the 100-dimensional prot2vecs to 2-dimensional vectors using the technique of PCA [305]. The results are shown in **Figure 5.2** and **Figure 5.3**, respectively. Note that **Figure 5.3 d-f)** were replaced with PF pairs whose protein members were better separated from each other in the mapped vector space.

**Table 5.1** Top rankings of the normalized average cosine distance between protein family pairs.

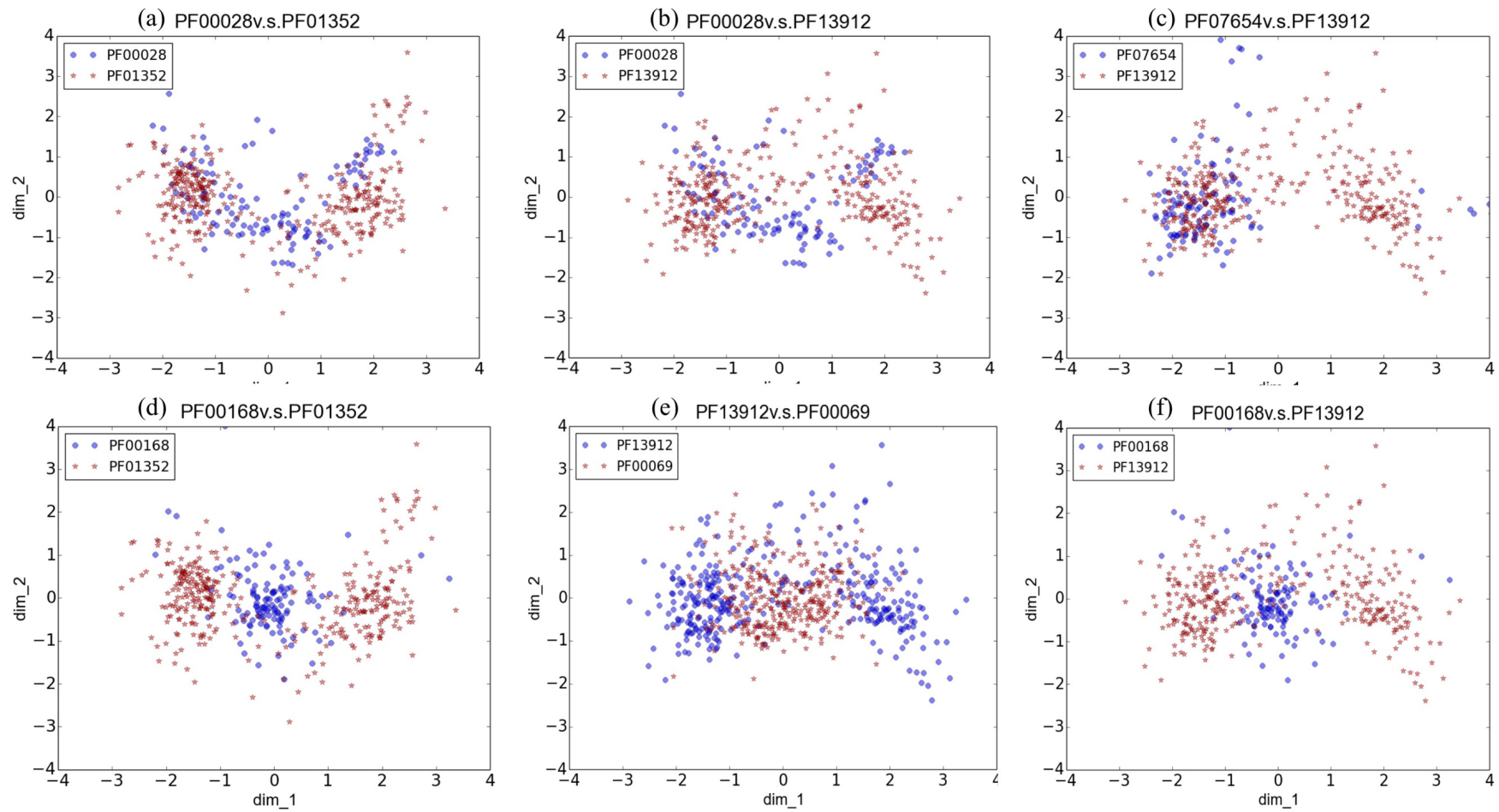
Top six PF pairs with min. cos distance				Top six PF pairs with max. cos distance			
	<i>PF_A</i>	<i>PF_B</i>	<i>Distance (Avg.)</i>		<i>PF_A</i>	<i>PF_B</i>	<i>Distance (Avg.)</i>
1	PF00096	PF13912	0.501		PF00028	PF01352	0.547
2	PF13912	PF01352	0.501		PF00028	PF13912	0.546
3	PF00096	PF01352	0.501		PF07654	PF13912	0.546
4	PF00041	PF07679	0.502		PF07654	PF01352	0.545
5	PF00169	PF00168	0.504		PF00001	PF01352	0.542
6	PF00595	PF00169	0.505		PF00096	PF00028	0.542

PF00096	Zinc finger
PF13912	C2H2-type zinc finger
PF01352	Kruppel associated box
PF00041	Fibronectin type III domain
PF07679	Immunoglobulin I-set domain
PF00001	Rhodopsin-like receptors

PF00169	PH domain
PF00168	C2 domain
PF00595	PDZ domain
PF00028	Cadherin domain
PF07654	Immunoglobulin C1-set domain
PF00069	Protein kinase domain



**Figure 5.2** Feature vector analysis for the top six PF pairs with minimum cosine distances.



**Figure 5.3** Feature vector space analysis of PF pairs with maximum cosine distances.

According to the results in **Figure 5.2**, the top three PF pairs with minimum cosine distances formed a closed similarity loop, in which *Zinc finger* (PF00096) is similar to *C2H2-type zinc finger* (PF13912), *C2H2-type zinc finger* (PF13912) is similar to *Kruppel associated box* (PF01352), and *Kruppel associated box* (PF01352) is similarity to *Zinc finger* (PF00096). The PF pair *Zinc finger* and *C2H2-type zinc finger* had the minimum cosine distance according to **Table 5.1** and their protein members were shown to have overlapping distributions in the vector space according to **Figure 5.2** (a). Considering *C2H2-type zinc finger* is by far the best-characterised class of zinc fingers, the overlapped distributions of proteins in PF00096 and PF01352 implicitly indicated that functionally related protein families are geometrically closer in prot2vec-based vector space. In **Figure 5.2** (c), protein members in *Kruppel associated box* had an overlap with but less wide distribution than those in *Zinc finger*, which validated the results in [306] that *Kruppel associated box* represents in approximately a third of zinc finger proteins.

The other three PF pairs whose members are closely distributed are *Fibronectin type III domain* (PF00041) and *Immunoglobulin I-set domain* (PF07679), *C2 domain* (PF00168) and *Pleckstrin homology domain (PH domain)* (PF00169), and *PDZ domain* (PF00595) and *PH domain*. Here, the protein family *C2 domain*, *PH domain* and *PDZ domain* are all closely associated with membrane targeting and signalling activities. This therefore confirms the previous observation that functionally related protein families are geometrically close to each other in prot2vec-based vector space. Interestingly, the protein members of *Fibronectin type III domain* and *Immunoglobulin I-set domain* in to **Figure 5.2** (d) do not demonstrate similar functions. They are both protein families that can be commonly found in various organisms, which might explain their close cosine distance in the mapped feature vector space.

**Figure 5.3** shows the six PF pairs whose members were most distant by normalised cosine distance. According to **Figure 5.3** (a), the most distant protein families are *Cadherin* and *Kruppel associated box*, who have different biological functions. Here, *Cadherin domain* (PF00028) is a class of type-1 transmembrane proteins while the *Kruppel associated box* represents a category of transcriptional repression domains. According to **Table 5.1**, *Cadherin* was shown to be distant from another zinc finger-related protein family *C2H2-type zinc finger*, which is extremely common in mammalian transcriptional factors. In correspondence, the distributions of *Cadherin* and *C2H2-type zinc finger* protein members demonstrated concave and convex character respectively in **Figure 5.3** (b). **Figure 5.3** (c) shows another protein family that is distant from *C2H2-type zinc finger*, i.e. *Immunoglobulin C1-set domain* which represents domains resembling the antibody constant domain.

The three PF pairs shown in **Figure 5.3** (d)-(f) were manually selected to demonstrate functionally divergent protein families which were clustered into different areas in the prot2vec-based vector space. For example, in **Figure 5.3** (f), proteins in *C2H2-type zinc finger* and *Protein kinase domain* (PF00069) were clearly separated in the mapped feature vector space. Regarding biological functionality, protein members in *Protein kinase domain* are involved in post-translational modifications during which phosphate groups can be attached to other amino acids, while those in *C2H2-type zinc finger* function as interaction modules that bind to DNA, RNA, proteins, or other small molecules [70].

In summary, the normalised cosine distances calculated based on prot2vec indicated the functional relationship between protein families. The visualisation in prot2vec-based vector space analysis in **Figure 5.2** and **Figure 5.3** demonstrated the effectiveness of using prot2vec as a input feature vector for protein family prediction.

## 5.5 Comparison with existing protein representations

In this section, the prediction performance of  $\text{prot2vec}^{\text{inference}}$  as the input feature in protein family prediction is compared to that of the existing distributed representation  $\text{prot2vec}^{\text{add}}$  (ProtVec [193]), the physicochemical property-based input feature and the homology profile-based input feature.

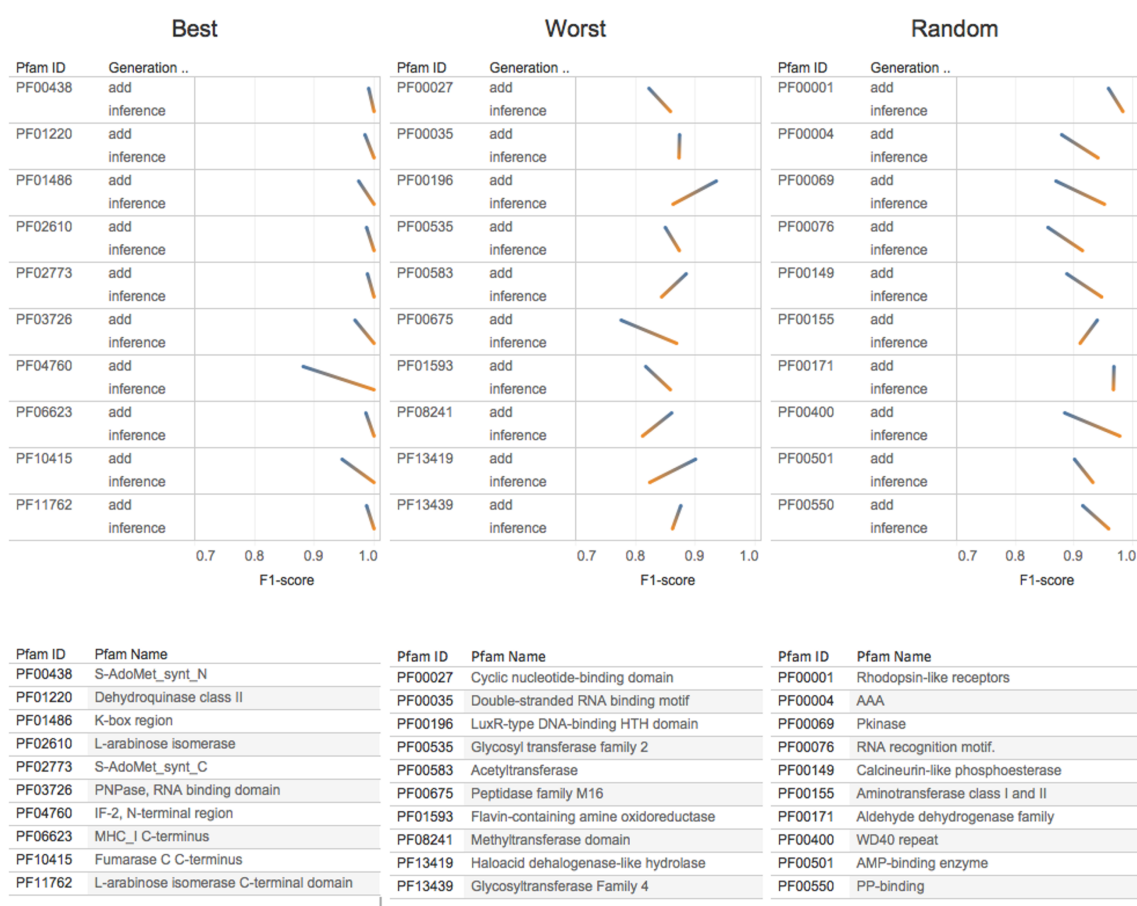
### 5.5.1 Comparison with existing distributed representations

As introduced in **Section 4.1.2**, the existing machine-generated protein feature vector  $\text{prot2vec}^{\text{add}}$  (termed *ProtVec* in its original paper [193]) was developed based on the word2vec model [307]. The vector presentation of protein sequences was constructed by adding up the pre-trained vectors of its biological words [308]. The key improvement of  $\text{prot2vec}^{\text{inference}}$  - when compared to  $\text{prot2vec}^{\text{add}}$  - is that it took into consideration the order of protein biological words or amino acid compositions when training the model for generating the distributed representations. In this investigation, we compare the prediction performance of models that respectively employed  $\text{prot2vec}^{\text{inference}}$  and  $\text{prot2vec}^{\text{add}}$  as the input feature in protein family prediction.

**Experimental design.** The prediction performance of  $\text{prot2vec}^{\text{add}}$ -based and  $\text{prot2vec}^{\text{inference}}$ -based PfamProt2vec was evaluated in 10-fold cross-validation tests, using *precision*, *specificity*, *recall* and the *F1-score* as the evaluation scores (see **Appendix A**). To ensure a profound statistical analysis of the prediction results, a subset of the UniProt-Pfam dataset was extracted by filtering out protein families with less than 100 protein members, resulting in 1,033 protein families for the cross-validation. For each of the 1,033 protein families, 10-fold cross-validation was performed, in which nine folds were used to train the SVM model while the remaining fold was used to validate the pre-trained SVM model. Since the negative sampling was performed for each protein family (see **Section 5.3**), both the training and validation were conducted based on balanced datasets. However, the issue of evaluation based on balanced datasets is that the performance of the predicative model tends to be overestimated. However, since both the  $\text{prot2vec}^{\text{inference}}$ -based PfamProt2vec and the  $\text{prot2vec}^{\text{add}}$ -based PfamProt2vec

were evaluated under the same experimental setting, it does not affect the comparison results.

**Result analysis.** The compared prediction performance of  $\text{prot2vec}^{\text{add}}$  and  $\text{prot2vec}^{\text{inference}}$ -based PfamProt2vec is shown in **Figure 5.4**. In order to provide an overview of the compared prediction performance of the two implementations of  $\text{prot2vecs}$ , we selectively demonstrate the results for 10 families in which the  $\text{prot2vec}^{\text{inference}}$ -based PfamProt2vec achieved the best F1-scores, 10 families in which the  $\text{prot2vec}^{\text{inference}}$ -based PfamProt2vec achieved the worst F1-scores, and 10 randomly selected families.



**Figure 5.4** Prediction performance of all protein families between  $\text{prot2vec}^{\text{inference}}$  and  $\text{prot2vec}^{\text{add}}$ .



According to **Figure 5.4**, among the 10 protein families in which the prot2vec<sup>inference</sup>-based PfamProt2vec achieved the best F1-scores, there were six *N-terminal or C-terminal domains*, one *RNA binding domain in PNPase*<sup>26</sup>, one *Dehydroquinase class I and II*<sup>27</sup>, one *K-box region* that is associated with SRF-type transcription factors, and one *L-arabinose isomerase*<sup>28</sup>. In summary, the prot2vec<sup>inference</sup>-based PfamProt2vec performed the best for predicting protein families that are involved in translation, transcription and corresponding enzyme catalytic chemical reactions.

Among the 10 protein families in which the prot2vec<sup>inference</sup>-based PfamProt2vec achieved the worst F1-scores, four of them performed better than the prot2vec<sup>add</sup>-based PfamProt2vec. For the 10 randomly selected protein families, the prot2vec<sup>add</sup>-based PfamProt2vec performed better for only one protein family *Aminotransferase class I and II*, which is a kind of pyridoxal-phosphate dependent enzyme. In addition, for the majority of the 10 randomly selected protein families, the prot2vec<sup>inference</sup>-based PfamProt2vec achieved higher precision, recall and F1 scores than those achieved by the prot2vec<sup>add</sup>-based PfamProt2vec.

**Table 5.2** demonstrates the compared average prediction performance between the prot2vec<sup>inference</sup>-based PfamProt2vec and the prot2vec<sup>add</sup>-based PfamProt2vec for protein families with more than 700 family members, with more than 200 family members, and with more than 100 family members, respectively, in terms of sensitivity (SE), specificity (SP) and precision (PR).

---

<sup>26</sup> PNPase is an enzyme that dismantles the RNA from the 3' end to the 5' end.

<sup>27</sup> Dehydroquinase class I and II is an enzyme that catalyses the 3-dehydroquinate chemical reaction

<sup>28</sup> L-arabinose isomerase is an enzyme that catalyses the L-arabinose chemical reaction

**Table 5.2** The average predictive performance of the  $\text{prot2vec}^{\text{inference}}$ -based PfamProt2vec and the  $\text{prot2vec}^{\text{add}}$ -based PfamProt2vec in different subsets of protein families.

	$\text{prot2vec}^{\text{inference}}$			$\text{prot2vec}^{\text{add}}$		
	<i>SE</i>	<i>SP</i>	<i>PR</i>	<i>SE</i>	<i>SP</i>	<i>PR</i>
PFs have $\geq 700$ members (Avg.)	<b>0.978</b>	0.981	<b>0.978</b>	0.881	<b>1.000</b>	0.911
PFs have $\geq 200$ members (Avg.)	<b>0.968</b>	0.967	<b>0.969</b>	0.825	<b>0.995</b>	0.878
PFs have $\geq 100$ members (Avg.)	<b>0.969</b>	0.969	<b>0.970</b>	0.832	<b>0.995</b>	0.881

From the results in **Table 5.2**, we made the following observations. Firstly, models which use more training examples are more likely to generate superior results. This is demonstrated by the superior performance achieved by protein families with more than 700 members in comparison to that of protein families with between 200 and 700 members. For example, with the  $\text{prot2vec}^{\text{inference}}$ -based PfamProt2vec, the average sensitivity score increased from 0.968 to 0.978 when the number of training examples increased from  $\geq 100$  to  $\geq 700$ . The average sensitivity of all protein families was 0.969, a score in between the previous two. Since there are more protein families with more than 100 members than protein families with more than 700 members. Secondly, the  $\text{prot2vec}^{\text{inference}}$ -based PfamProt2vec achieved superior average performance over different subsets of protein families, compared to the performance achieved by the  $\text{prot2vec}^{\text{add}}$ -based PfamProt2vec. This advantage in prediction performance was demonstrated in terms of sensitivity and precision. It indicates that models with  $\text{prot2vec}^{\text{add}}$  as the input feature vector were more likely to make false positive predictions, resulting in high specificity scores and relatively low sensitivity and precision scores.

In summary, the above results demonstrate that, for protein family prediction, where  $\text{prot2vecs}$  are used directly as the input feature vector, the  $\text{prot2vec}^{\text{inference}}$ -based PfamProt2vec outperformed the existing distributed representation of protein sequences the  $\text{prot2vec}^{\text{add}}$ -based PfamProt2vec.

### 5.5.2 Comparison with physiochemical properties-based features

To further validate the performance of prot2vec as the input feature for protein family prediction, we conducted a comparison between the performance of the PfamProt2vec based on prot2vec<sup>inference</sup> and the SVM-Prot 2016 method based on feature vectors generated from protein physicochemical properties. Note that the SVM-Prot 2016 was proposed to predict protein functional families, where each functional family was considered a collection of multiple protein families.

The compared input feature vectors include three protein descriptors, including composition (C), transition (T) and distribution (D), each of which was calculated from multiple protein physicochemical properties. The 13 manually selected protein physicochemical properties include amino acid composition, polarity, hydrophobicity, surface tension, charge, normalized Van der Waals volume, polarizability, secondary structure, solvent accessibility, molecular weight, solubility, number of hydrogen bond donors in side chain, and number of hydrogen bond acceptor in side chain [295].

**Experimental design.** In order to obtain the exactly identical dataset based on which the SVM-Prot 2016 was trained and tested, we constructed the training, testing and independent test dataset strictly according to the procedures described in SVM-Prot 2016 [295].

- Firstly, we searched the 10 representative functional families (denoted with GO [249] identifications) in the Swiss-Prot/UniProt database [39], resulting in protein members for each of the functional family.
- Secondly, we obtained for each functional family  $FF_i$  a Pfam [70] identification list  $L_i = (PF1_1, PF2, \dots, PF_m)$ , where each protein family  $PF_j$  had at least one protein member in  $FF_i$ .
- Thirdly, we iteratively put the protein members of each functional family  $FF_i$  to its training, testing and independent test dataset, as positive examples.

- Fourthly, for each functional family  $FF_i$ , we iteratively put 3 protein members from protein families that do not belong to  $L_i$  to its training, testing and independent test dataset, as negative examples.

**Table 5.3** provides the detailed statistics of the constructed datasets for 10 representative protein functional families.

**Table 5.3** Statistics of the constructed training, test and independent test datasets for 10 representative protein functional families.

Go Name	Training Set		Test Set		Independent Test Set	
	<i># pos</i>	<i># neg</i>	<i># pos</i>	<i># neg</i>	<i># pos</i>	<i># neg</i>
GO:0051693	115	12,245	57	10,806	58	9,649
GO:0006310	3,947	11,776	1,973	10,349	1,974	9,239
GO:0006281	7,469	11,577	3,734	10,133	3,735	9,036
GO:0016645	1,021	12,234	510	10,756	512	9,656
GO:0016785	120	12,295	60	10,849	61	9,703
GO:0016846	322	12,267	161	10,843	161	9,688
GO:0016854	1,141	12,244	570	10,800	571	9,651
GO:0003746	1,957	12,264	978	10,802	979	9,675
GO:0004930	1,618	12,189	809	10,741	810	9,601
GO:0008289	2,168	11,672	1,084	10,220	1,085	9,072

**Table 5.4** Performance comparison between PfamProt2vec and SVM-Prot 2016 for 10 representative families.

Family ID	Functional family name	PfamProt2vec			SVM-Prot 2016		
		<i>SE</i>	<i>SP</i>	<i>PR</i>	<i>SE</i>	<i>SP</i>	<i>PR</i>
GO:0051693	Actin capping	0.298	<b>1.000</b>	<b>1.000</b>	<b>0.951</b>	0.999	0.933
GO:0006310	DNA recombination	0.587	0.973	0.868	<b>0.857</b>	0.974	<b>0.921</b>
GO:0006281	DNA repair	0.719	0.886	0.779	<b>0.887</b>	0.968	<b>0.859</b>
GO:0016645	EC1.5 Oxidoreductases (CH-NH donors)	0.575	<b>1.000</b>	<b>0.994</b>	<b>0.586</b>	0.996	0.661
GO:0016785	EC2.9 Transferases (selenium-containing)	0.721	<b>1.000</b>	<b>1.000</b>	<b>0.960</b>	0.999	0.993
GO:0016846	EC2.9 Transferases (selenium-containing)	<b>0.615</b>	<b>1.000</b>	<b>1.000</b>	0.603	0.999	0.833
GO:0016854	EC5.1 Racemases and Epimerases	<b>0.669</b>	0.993	<b>0.886</b>	0.530	0.994	0.539
GO:0003746	Elongation factor activity	0.921	0.996	0.972	<b>0.975</b>	0.999	0.988
GO:0004930	GPCR	0.877	<b>0.990</b>	0.905	<b>0.956</b>	0.981	0.945
GO:0008289	Lipid-binding	0.143	<b>1.000</b>	<b>0.987</b>	<b>0.844</b>	0.999	0.934

\* The better performance in terms of SE, SP and PR were highlighted as bold.

**Results and analysis.** In **Table 5.4**, the performance of PfamProt2vec and SVM-Prot 2016 was evaluated and compared in terms of sensitivity (SE), specificity (SP) and precision (PR). Please refer to **Appendix A** for detailed description of evaluation measurements.

PfamProt2vec achieved better performance for 6 out of 10 families in terms of the specificity and precision, but worse performance in 8 out of 10 families in terms of sensitivity. This suggests that SVM tends to make more cautious predictions by using prot2vec as the feature vector. According to our observation, the only family for which PfamProt2vec achieved overall better performance is *EC2.9 Transferases (selenium-containing)*, with a combined score of 0.669 (SE), 1.0 (SP) and 1.0 (PR), while SVM-Prot 2016 achieved better performance for *DNA recombination*, *DNA repair* and *Elongation factor activity*. We conclude that PfamProt2vec achieved comparable performance with SVM-Prot 2016 in terms of overall performance, but with better performance for application scenarios where prediction accuracy is more pertinent.

### 5.5.3 Comparison with homology profile-based features

The generation of homology profiles, such as the PSSMs [99] and the HMM profiles [87], relies on protein sequence alignment algorithms that are designed based on biological heuristics. In contrast, the distributed representation prot2vec is generated directly from protein sequences using unsupervised machine learning algorithms [169, 289] that are purely data-driven. Therefore, the former represents the most relevant feature for protein family prediction according to biological heuristics while the latter represents the data-driven representation that involves no domain knowledge.

In this investigation, we compare the performance of the data-driven representation prot2vec<sup>inference</sup> to that of the biologically driven representation HMM profiles, as the input feature for protein family prediction.

**Experimental design.** Since the UniProt-Pfam dataset was extracted from the Pfam database, a protein family classification system that was constructed based on the

HMMER algorithm, we used the UniProt-InterPro dataset (see **Section 5.3.2**) as the benchmark dataset to evaluate the performance of PfamProt2vec and HMMER 3.1.

We conducted performance evaluation for proteins in *Escherichia coli*<sup>29</sup> (*E. coli*) at the proteomic level. Here, *E. coli* represents one of the most researched and well-annotated proteomes [309-311]. According to the Swiss-Prot/UniProt database [39], the *E. coli* proteome has 4,309 proteins all of which are manually reviewed. We compared the prot2vec<sup>inference</sup>-based PfamProt2vec with the HMM profile-based HMMER 3.1, by predicting the 1,461 protein families in the constructed UniProt-InterPro dataset for 4,309 *E. coli* proteins.

In this investigation, we also considered using different thresholds for distinguishing positive predictions from negative predictions. In most cases, the cut-off threshold selected for the output scores are not applicable across multiple methods, because different methods produce scores within different ranges. One solution is to select cut-off thresholds for one of the evaluation measurement scores, so that different methods have the same level of one particular measurement. For example, a cut-off threshold of 0.1 for *false positive rate* (FPR) indicates that in every 10 negative examples one will be predicted as false positive. The cut-off threshold 0.99 for the *true positive rate* (TPR or sensitivity) means that in every 100 positive examples 99 will be recognised. With these cut-off thresholds selected, different prediction methods are evaluated in the same level in terms of one particular measurement, even if they produce different ranges of output scores.

**Result analysis.** For each of the 1,461 protein families in the UniProt-InterProt dataset, we calculated the precision (PR), sensitivity (SE), and balanced accuracy (BACC) (see **Appendix A**) scores respectively for PfamProt2vec and HMMER 3.1.

---

<sup>29</sup> *Escherichia coli* is a coliform bacterium of the genus *Escherichia* that is commonly found in the lower intestine of warm-blooded organisms ([https://en.wikipedia.org/wiki/Escherichia\\_coli](https://en.wikipedia.org/wiki/Escherichia_coli)).



**Table 5.5** reports the average PR, SE and BACC scores over 1,461 protein families with the false positive rate (FPR) thresholds set to 0.1, 0.01, 0.001 and 0.0001, respectively.

**Table 5.5** Performance comparison between PfamProt2vec and HMMER 3.1 for protein family prediction in corresponding families.

FPR threshold	PfamProt2vec			HMMER 3.1		
	<i>PR</i>	<i>SE</i>	<i>BACC</i>	<i>PR</i>	<i>SE</i>	<i>BACC</i>
0.0001	1.000	0.786	0.893	1.000	<b>0.950</b>	<b>0.975</b>
0.001	<b>0.574</b>	0.851	0.926	0.564	<b>0.989</b>	<b>0.994</b>
0.01	<b>0.488</b>	0.916	0.957	0.047	<b>0.997</b>	<b>0.995</b>
0.1	<b>0.262</b>	0.977	0.979	0.014	<b>0.998</b>	<b>0.982</b>

According to **Table 5.5**, the performance of PfamProt2vec achieved superior precision scores, while HMMER 3.1 achieved superior sensitivity scores for the four FPR thresholds. For performance with FPR threshold set to 0.01 and 0.1, the precision of PfamProt2vec was kept at a score of 0.488 and 0.262 while, at the same time, reasonably satisfying sensitivities were achieved at a score of 0.916 and 0.977, respectively. In comparison, HMMER 3.1 achieved precision scores of 0.047 and 0.014 for FPR threshold 0.01 and 0.1.

The results in **Table 5.5** demonstrated the following conclusions. Firstly, in terms of the balanced accuracies, HMMER 3.1 performed better than PfamProt2vec for all FPR thresholds. This indicates that HMMER 3.1 is better at dealing with imbalanced data in protein family prediction. Secondly, in terms of precision, HMMER 3.1 achieved scores below 0.1 for FPR threshold 0.1 and 0.01, while PfamProt2vec managed to achieve scores above 0.2 even with high false positive rate thresholds. In contrast, the performance of PfamProt2vec and HMMER 3.1 for FPR threshold 0.001 and 0.0001 is very close. This suggests that HMMER 3.1 was optimised to perform well under situations with low FPR thresholds, which represents the requirement of most application

scenarios. Finally, in terms of the sensitivity score, HMMER 3.1 consistently outperformed PfamProt2vec in all FPR thresholds. This demonstrates that HMMER 3.1 performed better than PfamProt2vec in covering more positive samples in each protein family, even with low FPR thresholds.

## 5.6 Investigation of potential factors that affect the prediction performance of PfamProt2vec

When prot2vec is used as the feature vector for protein family prediction, multiple factors exist that may affect the prediction performance of predictive models for respective protein families. In this section, we perform comprehensive experiments to investigate the factors that may affect the prediction performance of PfamProt2vec.

### 5.6.1 The effect of the organism of training examples

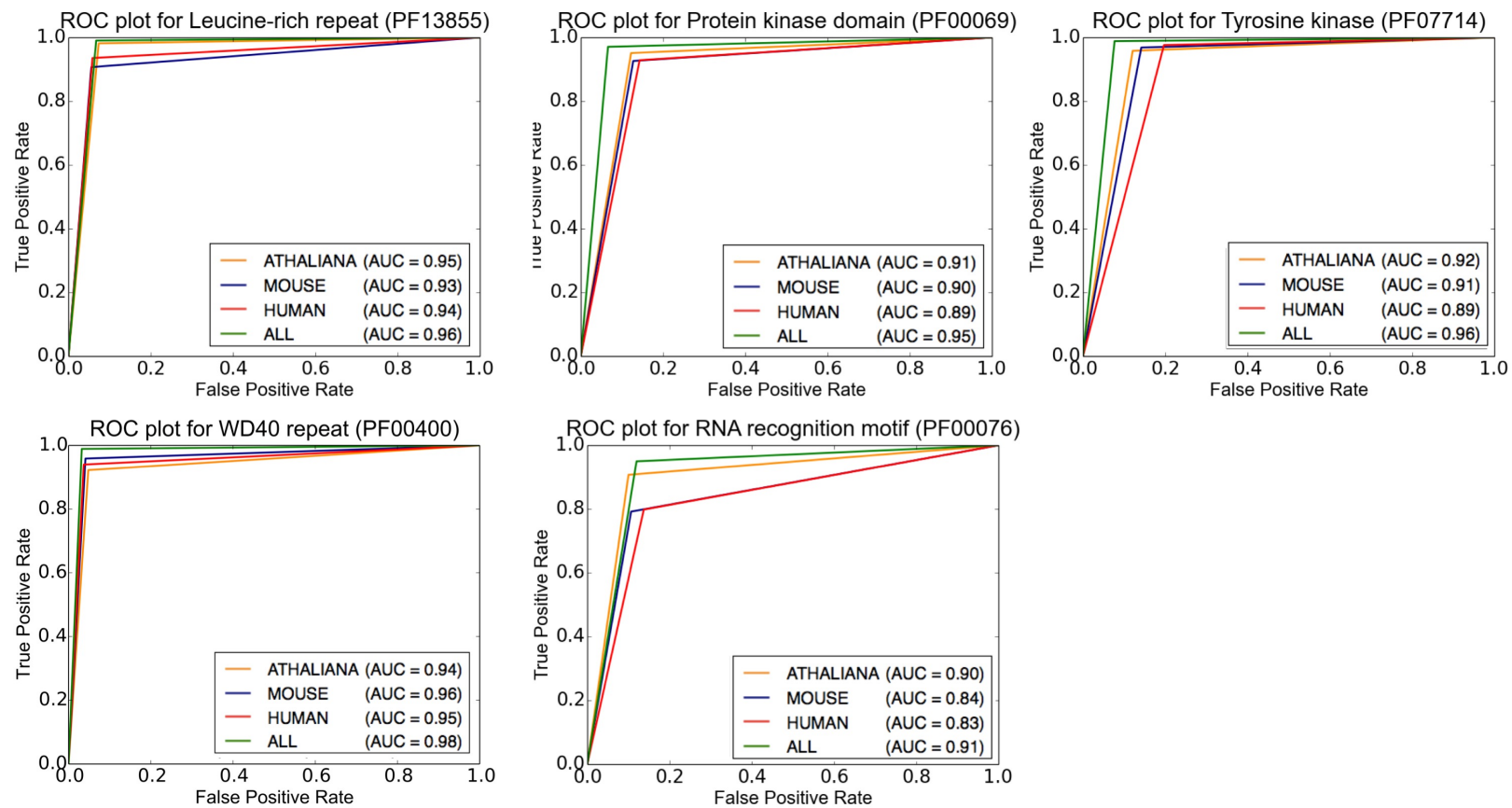
In this investigation, we analyse the prediction performance of PfamProt2vec models that are trained with protein members from different organisms. We applied the prot2vec to predict members of protein families that were shared across three most populated organisms including *Human*, *Mouse* and *A. thaliana*<sup>30</sup>. For proteins in these three organisms, we identified five common protein families from the 9,337 protein families in the UniProt-Pfam dataset, each of which had at least 100 protein members in each of the three organisms. They are *Leucine-rich repeats* (PF13855), *Protein kinase domain* (PF00069), *Tyrosine kinase* (PF07714), *RNA recognition motif* (PF00076) and *WD40 repeat* (PF00400). By training organism-specific PfamProt2vec models, we demonstrate the correlation between the prediction performance of PfamProt2vec and the source organism of the training examples.

---

<sup>30</sup> Arabidopsis thaliana (*A. thaliana*): [https://en.wikipedia.org/wiki/Arabidopsis\\_thaliana](https://en.wikipedia.org/wiki/Arabidopsis_thaliana). It is a small flowering plant native to Eurasia and Africa. It is a popular model organism for plant biology research.

**Experimental design.** For each family, we extracted their members in organism Human, organism Mouse, organism *A. thaliana* and all organisms, as positive examples, respectively. We subsequently constructed corresponding datasets by performing negative sampling to ensure a positive to negative ratio of 1:1, resulting in four datasets termed as HUMAN, MOUSE, ATHALIANA and ALL, respectively. On each of the four datasets, 10-fold cross-validation test was performed, based on which each of the corresponding organism-specific PfamProt2vec models was evaluated in terms of the ROC curve and the AUC score (see **Appendix A**).

**Results analysis.** In **Figure 5.5**, based on the cross-validation results, we plotted the ROC curves with AUC scores of four organism-specific PfamProt2vec models (trained based on the dataset ATHALIANA, MOUSE, HUMAN and ALL, respectively) for each of the five protein families.



**Figure 5.5** Prediction performance of Prot2vecDM for five common protein families shared across human, mouse, *A. thaliana* and all organisms.

As reflected by the largest AUC scores in **Figure 5.5**, for all five protein families, PfamProt2vec consistently performed better with models trained using examples from all organisms than those trained using examples from any of the individual organisms. For example, for protein family *RNA recognition motif*, the AUC score was increased from 0.83 (for HUMAN) to 0.91 (for ALL) when protein samples from all organisms were used for training the model. To understand the effect of protein members from other organisms on the prediction performance in HUMAN, we compared the numbers of false positives and false negatives predicted by models trained on HUMAN with those trained on ALL. We found that the numbers of false positives and false negatives were respectively reduced by 28 and 34 out of 407 predictions. These results suggest that the knowledge learned from one organism can be transferred to other organisms and, as a potentially useful strategy, protein members from different organisms should be combined to achieve better predictive performance for protein family prediction.

### 5.6.2 The effect of functional categories of protein families

In this section, the effect of functional categories and the number of training examples on the prediction performance of PfamProt2vec was investigated.

**Experimental design.** To investigate the effect of functional categories and the number of training examples, we performed protein family prediction for 25 human protein families extracted from the UniProt-Pfam dataset. In this investigation, only human protein members were extracted as positive examples for each protein family; correspondingly, an equal number of negative examples were sampled from the remaining human proteins that do not belong to the protein family. Based on 10-fold cross-validation tests, the prediction performance of PfamProt2vec for each protein family was evaluated in terms of *sensitivity (SE)*, *specificity (SP)* and *precision (PR)* (see **Appendix A**).









The tested 25 protein families were grouped into 8 functional categories, *i.e.* *zinc finger-related families* (e.g. PF00096, PF13912 and PF01352), *transmembrane protein*

*families (e.g. PF00028 and PF00520), intracellular signalling and targeting-related families (e.g. PF00001, PF00071, PF00168, PF00169 and PF00595), enzymes (e.g. PF00089, PF00069 and PF07714), extracellular protein family (e.g. PF00041), immunoglobulins domains (e.g. PF07654, PF07679 and PF07686), DNA binding-related families (e.g. PF00046 and PF00010) and structural motifs (e.g. PF00400, PF13855, PF00271, PF12796, PF00076 and PF00651).*

**Result analysis.** The number of positive examples and the corresponding prediction performance of PfamProt2vec for 25 human protein families are reported in the following table. The results were ordered according to the largest-to-smallest ranking in terms of precision scores. The results of protein families belonging to the same functional categories are highlighted with the same colour. For example, results of all zinc finger related protein families are highlighted with the green colour.

**Table 5.6** Prediction performance of PfamProt2vec for predicting 25 human protein families according to eight functional categories

PF ID	PF Name	# proteins	SE	SP	PR
PF00096	Zinc finger	533	0.994	0.992	0.995
PF00028	Cadherin	114	0.987	0.992	0.988
PF13912	C2H2-type zinc finger	332	0.983	0.985	0.984
PF01352	Kruppel associated box	361	0.972	0.995	0.974
PF00089	Trypsin	119	0.970	0.977	0.974
PF00001	Rhodopsin-like receptors	285	0.965	0.947	0.965
PF00046	Homeodomain fold	222	0.943	0.958	0.945
PF07654	Immunoglobulin C1-set domain	144	0.937	1.000	0.948
PF00400	WD40 repeat	217	0.931	0.914	0.929
PF13855	Leucine-rich repeat	171	0.929	0.974	0.934
PF00520	Ion channel family	106	0.921	0.842	0.937
PF00069	Protein kinase domain	350	0.887	0.852	0.888
PF00071	Ras subfamily	137	0.882	0.941	0.882
PF07686	Immunoglobulin V-set domain	281	0.881	0.904	0.883
PF00041	Fibronectin type III domain	135	0.881	0.838	0.881
PF07714	Tyrosine kinase	127	0.877	0.790	0.885
PF07679	Immunoglobulin V-set domain	136	0.869	0.819	0.884
PF00169	Pleckstrin homology domain	178	0.859	0.779	0.867
PF12796	Ankyrin repeat	227	0.853	0.844	0.851
PF00076	RNA recognition motif.	204	0.835	0.863	0.837
PF00010	Basic helix-loop-helix	107	0.833	0.877	0.835
PF00271	Helicase conserved C-terminal domain	106	0.771	0.580	0.832
PF00168	C2 domain	124	0.769	0.634	0.788
PF00651	BTB/POZ domain	129	0.766	0.676	0.778
PF00595	PDZ domain	129	0.759	0.669	0.763

	Zinc finger-related families		Intercellular signaling and targeting related families
	Transmembrane protein families		Enzymes
	Immunoglobulins families		DNA-binding related families
	Structural motifs		Extracellular proteins families

According to **Table 5.6**, all zinc finger-related families had more than 300 protein members. They were predicted with the best prediction performance among all 8 functional groups. It confirmed the vector space analysis in **Figure 5.3** where zinc finger-related protein families were clearly separated from other protein families in the prot2vec-based vector space. The second-best performance was achieved for the *Cadherin* family with a sensitivity of 0.987 and a precision of 0.988, even though it had only 114 protein members. Similarly, a reasonably satisfying performance was achieved for the *Ion channel family*, which is a transmembrane-related protein family with only 106 positive examples. In contrast, for family *Protein kinase domain* which have 350 protein members, PfamProt2vec only achieved a sensitivity of 0.887. This is the only protein family with more than 300 members for which PfamProt2vec achieved a sensitivity less than 0.9. These results indicate that for specific functional groups, for example, the transmembrane-related families, a satisfying prediction performance could be achieved even with a limited number of training examples; while for other protein families, such as the *Protein kinase family*, more training examples do not guarantee better prediction performance.

The positive effect of incorporating larger numbers of training examples was demonstrated in intracellular signalling and targeting-related families. For protein families related to intracellular signalling and targeting activities, four out of five were predicted with a sensitivity of less than 0.9. These families had fewer positive examples, from between 124 and 285. Among these, the only protein family for which PfamProt2vec achieved a sensitivity of 0.965 had a relatively larger number of 285 positive examples.

Regarding protein families that contain structural motifs, PfamProt2vec achieved a sensitivity above 0.9 for only two protein families, *i.e.* *WD40 repeat* and *Leucine-rich repeat*. For this functional group, there was no clear association between the performance and the number of training examples. This might be explained by the presence of diverse structural motifs with different physicochemical properties, structures and biological functions.



Regarding the number of training examples, when using more than 200 examples 6 out of 10 families achieved a sensitivity of above 0.9, whereas a sensitivity of below 0.9 was achieved in 10 out of 14 samples when fewer than 200 training examples were used. In terms of functional categories, PfamProt2vec achieved better performance for specific functional groups, including zinc finger-related protein families and transmembrane-related protein families, but tended to perform worse for intracellular signalling and targeting-related families and structural motifs.

### 5.6.3 The effect of eight manually selected factors

In **Section 5.5.3**, we compared the prediction performance of PfamProt2vec and HMMER 3.1, from which we observed the superior performance of the latter. For the purpose of better understanding the performance difference, we summarised the difference between PfamProt2vec and HMMER 3.1 in terms of methodology in **Table 5.7**, based on which we extracted 8 factors that may affect the performance of PfamProt2vec in protein family prediction.

**Table 5.7** Differences between HMMER 3.1 and PfamProt2vec in terms of methodology.

	<b>PfamProt2vec</b>	<b>HMMER 3.1</b>
Models	Support vector machines where each protein family corresponds to a pre-trained SVM model for binary classification.	Hidden Markov models where each protein family corresponds to a pre-trained HMM model for sequence matching.
Features	Prot2vecs generated based on sequential contexts within proteins	Homology profiles generated based on multiple sequence alignment
Granularity	Sequence-level where the prediction is conducted based on the feature extracted for whole protein sequences.	Domain level prediction where the prediction is conducted based on the matching status of sequence segments in protein sequences.

Since the performance of models for different protein families achieved by PfamProt2vec are dramatically different, herein we modelled the correlation between the performance achieved by PfamProt2vc and the 8 manually selected factors, including

1. *The sequential diversity within each family.* The sequential similarity is one of the features of proteins that belong to the same protein family. Since models in

PfamProt2vec were trained based on feature vectors (prot2vecs) generated from sequential patterns, therefore PfamProt2vec is hypothetically more likely to achieve inferior performance in protein families with dissimilar protein sequences.

2. *The homologous relation within each family.* Protein families refer to groups of proteins that share the same evolutionary origin. Therefore, close homology relation is another feature of proteins that belong to the same protein family. Since PfamProt2vec did not involve any evolutionary-based features while models in HMMER 3.1 were trained purely based on homologous information, it would be interesting to compare the effect of homologous relation on the performance of both of these prediction methods.
3. *The average length of annotated sequence segments in each family.* For each protein family, its annotated sequence segments in any protein do not necessarily occupy the whole protein, where shorter annotated sequences may be more likely to co-exists with other family/domains within the same protein sequence. Since models in PfamProt2vec take as input the vector representation (prot2vec) generated for whole protein sequences, it is hypothetically more difficult for PfamProt2vec to predict members for families with shorter annotated sequence segments. However, for HMMER 3.1, the sequence matching was performance residue-by-residue, its prediction performance may suffer less from this factor.
4. *The variation of lengths of annotated sequences in each family.* For any protein family, the lengths of its annotated sequence segments in different proteins may vary slightly or dramatically, depending on the evolutionary history. However, annotated sequences of dramatically various lengths may reflect less similarity among the annotated sequences of a protein family, for which it is hypothetically more difficult for PfamProt2vec to achieve satisfying prediction performance.
5. *The number of member proteins in each family.* This statistic determines the number of training example for each protein family, where the involving of more training examples reduces the risk of over-fitting during model training. However,

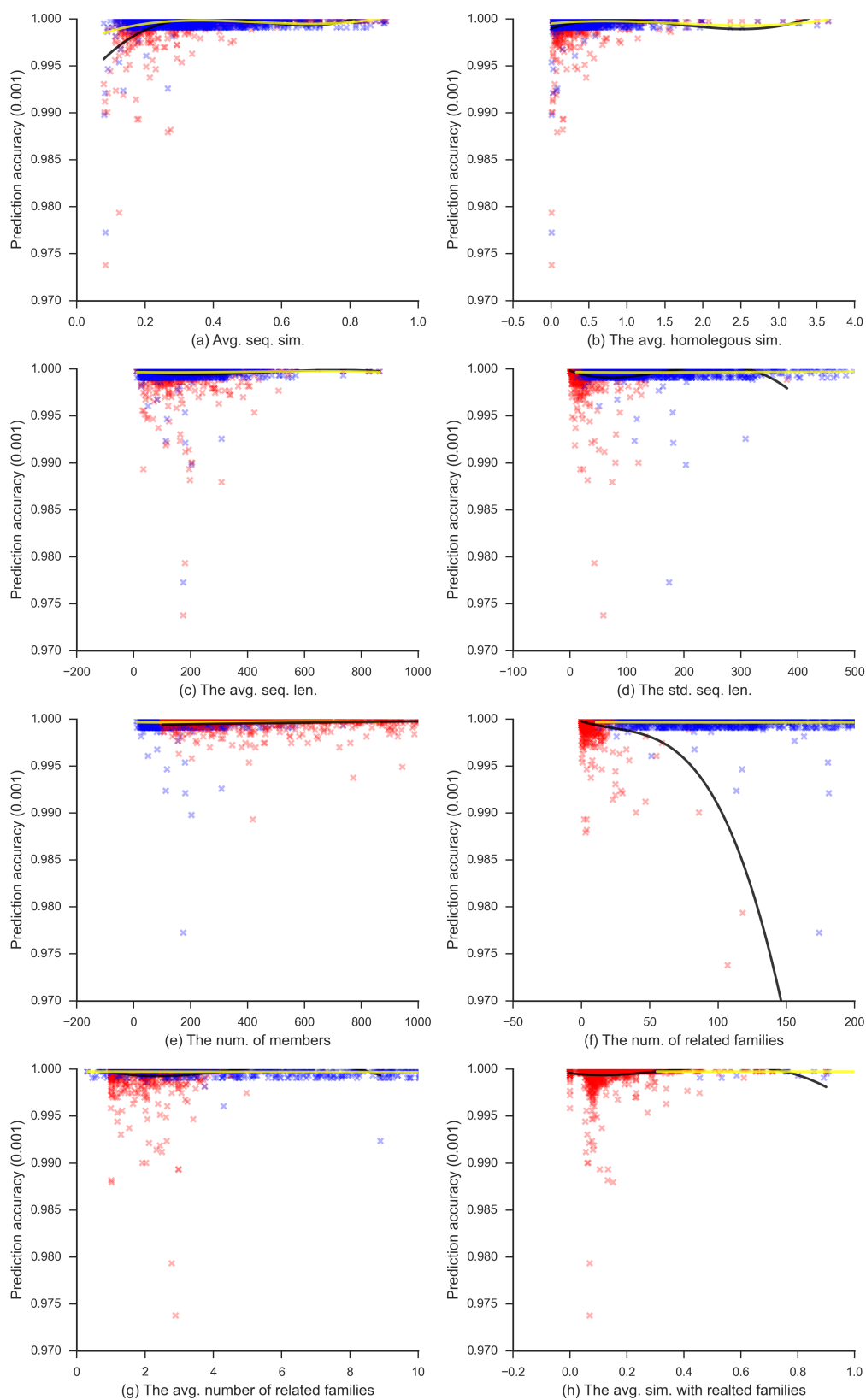
more protein members in a protein family may also mean a higher likelihood of sequentially divergent proteins, which makes it difficult for PfamProt2vec to achieve satisfying performance.

6. *The number of families sharing members with each family.* Since models in PfamProt2vec take as input the vector representation generated from whole protein sequences, it is hypothetically more likely for PfamProt2vec to make a false positive prediction for protein families overlapping with more other protein families. The other families that share members with a protein family A are referred to as the related families of the protein family A.
7. *The average number of protein families for protein members in each family.* This statistic represents the number of other protein families for each protein member in the protein family. Higher values indicate that proteins in this family are more likely to have more annotated families/domains within the same protein sequence. Therefore, PfamProt2vec is hypothetically more likely to achieve less satisfying performance.
8. *The average sequential similarity with families that share members with each family.* If the protein members of two protein families are sequentially similar to each other, it is more likely for PfamProt2vec to predict the member in one protein family as a false positive prediction in the other family. Therefore, PfamProt2vec is hypothetically more likely to achieve less satisfying performance in protein families with less similarity with their related families.

For each of the 8 above factors, we analysed its correlation with the prediction performance achieved by PfamProt2vec and HMMER 3.1, for each protein family.

**Figure 5.6** demonstrates the calculated correlation for the 8 factors in all of the 1,461 protein families in the UniProt-InterPro database, with the FPR threshold set to 0.001. Here, blue cross and red cross demonstrated the results of HMMER 3.1 and PfamProt2vec, respectively. While the yellow line and the black line represents the fitted

regression line plot, with a regression order of 3, for HMMER 3.1 and PfamProt2vec, respectively.



**Figure 5.6** The correlation between the eight potential factors and the prediction accuracy of PfamProt2vec (red) and HMMER 3.1 (blue).

The **effect of sequential diversity within each family**. In order to investigate the correlation between the prediction performance of PfamProt2vec and the sequential diversity within each protein family, we defined the following similarity measurement for any protein family  $A$ ,

$$Sim(A) = \frac{1}{N^2} \sum_{i=1}^N \sum_{j=1}^N cosine\_similarity(v_i, v_j) \quad [\text{Eq. 5.2}]$$

where  $N$  denotes the number of protein member in family  $A$ , and  $v_i$  and  $v_j$  denote the prot2vecs of the  $i$ -th and  $j$ -th protein member of family  $A$ . A smaller score of  $Sim(A)$  indicates that the sequences of protein in family  $A$  have a higher sequential diversity. This score measures the sequential similarity between protein members within the same protein family, by calculating cosine similarities between prot2vecs of any pair of two proteins. The basic assumption of this measurement is that proteins that are closer in the prot2vec-based vector space are more likely to be sequentially similar to each other.

Results in **Figure 5.6** (a) demonstrated that PfamProt2vec performed worse for protein families in which protein members are less sequentially similar to each other, while the prediction performance of HMMER 3.1 is less affected by this factor. Worse performance of both PfamProt2vec and HMMER 3.1 was achieved in protein families with lower sequential similarities. However, PfamProt2vec achieved accuracy scores less than 0.998 in more protein families than HMMER 3.1 did. Since HMMER 3.1 is based on homology profiles, it is less sensitive to the sequential similarity within each protein families. More specifically, even the proteins in a protein family demonstrated divergent sequential patterns, as long as their homologous profiles are similar, its prediction performance will not be negatively affected. Results in **Figure 5.6** (a) explains the superior performance of HMMER 3.1 compared with that of the PfamProt2vec.

**The homologous relation within each family**. The homologous relation between proteins has been represented in multiple ways including the PSSMs and the phylogenetic trees. Since the generation of PSSMs is more time-consuming, here we

use the tree distance between proteins in a phylogenetic tree as the measurement for homologous relation.

We calculated the phylogenetic-tree distance in the following steps. Firstly, we performed multiple sequence alignment for protein members in each protein family, using Clustal-Omega 1.2.4<sup>31</sup>. Secondly, we generated the phylogenetic tree for each protein family based on the generated multiple sequence alignment from the first step, using the online web service T-REX [312]. Thirdly, we calculated the average homologous similarity for each protein family as follows,

$$Phylo\_sim(A) = \frac{1}{N^2} \sum_{i=1}^N \sum_{j=1}^N \frac{1}{phylo\_dist_A(p_i, p_j)} \quad [\text{Eq. 5.3}]$$

where  $N$  denotes the number of protein members in family  $A$ , and  $phylo\_dist(p_i, p_j)$  denotes the tree distance between the  $i$ -th and the  $j$ -th protein member of family  $A$ . The tree distance between protein  $p_i$  and  $p_j$  was calculated using the python package BioPy<sup>32</sup>. Larger  $Phylo\_sim(A)$  indicates that proteins in family  $A$  are evolutionarily related to each other.

Results in **Figure 5.6 (b)** demonstrated that PfamProt2vec achieved better performance for protein families with lower homologous similarities than for protein families with lower sequence similarities. Despite the slightly better performance achieved by HMMER 3.1 in protein families with low homologous similarities, the performance of PfamProt2vec does not seem to be affected by the homologous similarity within each protein family. The fitted regression lines of HMMER 3.1 (yellow) and PfamProt2vec (red) are similar across all values in terms of the order of derivatives. However, in terms of the scattered plot, PfamProt2vec (red cross) achieved inferior performance in more protein families with low homologous similarity than HMMER 3.1 did.

---

<sup>31</sup> Clustal-Omega: <http://www.clustal.org/omega/>

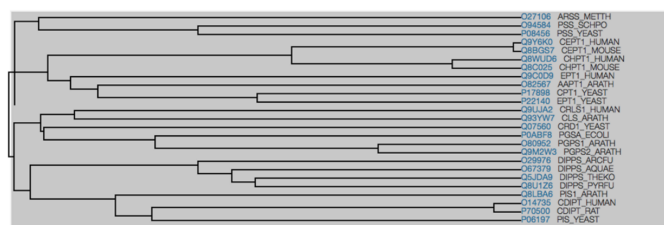
<sup>32</sup> BioPy: [https://figshare.com/articles/biopy\\_a\\_Library\\_for\\_Phylogenetic\\_Exploration/761224](https://figshare.com/articles/biopy_a_Library_for_Phylogenetic_Exploration/761224)

We further visualized the phylogenetic trees of protein families for which PfamProt2vec achieved the best the worst prediction accuracies, including *the Glycoside hydrolase, family 34* (IPR001860), *Malate synthase* (IPR001465), *Photosystem antenna protein-like* (IPR000932), *Glycosyltransferase 2-like* (IPR001173), *Phosphatidic acid phosphatase type 2/haloperoxidase* (IPR000326), *CDP-alcohol and phosphatidyltransferase* (IPR000462). **Table 5.8** and **Figure 5.7** demonstrate the protein family information and their corresponding phylogenetic trees, respectively.

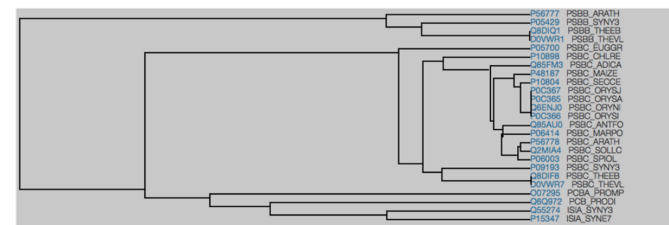
**Table 5.8** Protein families with the best and the worse prediction accuracies achieved by PfamProt2vec.

InterPro ID	PR	SE	SP	ACC	F1	#pos	#pred_pos
IPR000462	0.007	1.000	0.895	0.894	0.013	3	456
IPR000326	0.009	1.000	0.899	0.899	0.018	4	436
IPR001173	0.021	0.9	0.901	0.901	0.04	10	433
IPR000932	1.000	1.000	1.000	1.000	1.000	0	0
IPR001465	1.000	1.000	1.000	1.000	1.000	2	2
IPR001860	1.000	1.000	1.000	1.000	1.000	0	0

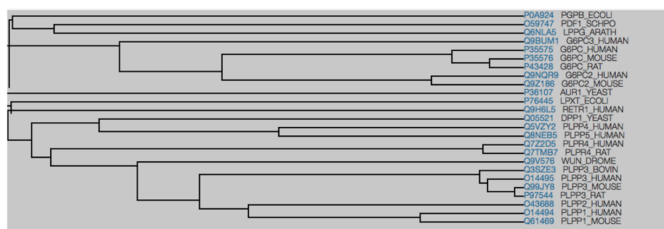




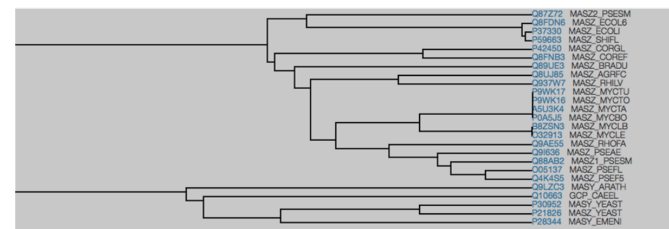
(a) CDP-alcoholphosphatidyltransferase (IPR000462)



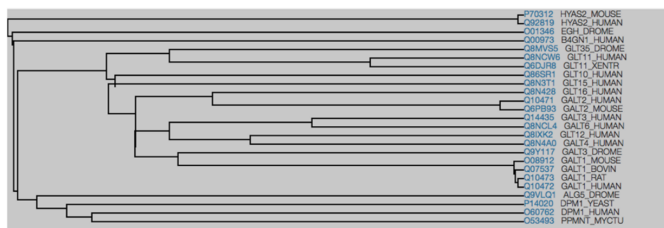
(b) Photosystem antenna protein-like (IPR000932)



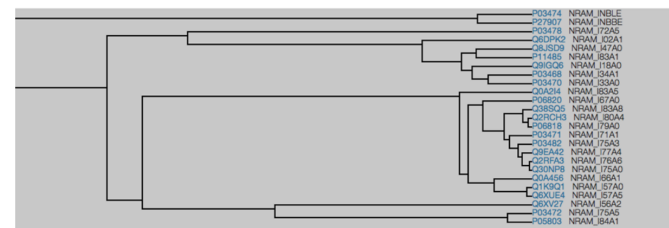
(c) Phosphatidic acid phosphatase type 2/haloperoxidase (IPR000326)



(d) Malate synthase (IPR001465)



(e) Glycosyltransferase 2-like (IPR001173)



(f) Glycoside hydrolase, family 34 (IPR001860)

**Figure 5.7** Phylogenetic trees of the protein families with the worse prediction accuracies (left) and the protein families with the best prediction accuracies (right).

Results in **Table 5.8** and **Figure 5.7** showed that phylogenetic trees of protein families with the best and the worst performance achieved by PfamProt2vec demonstrated different tree structures. In phylogenetic trees, any two tree nodes are considered to be evolutionarily distant from each other if their split point is closer to the root of the tree<sup>33</sup>. Therefore, according to the left three subgraphs in **Figure 5.7**, protein members are more evolutionarily distant from each other in protein families with the worse performance. However, in the right three subgraphs, for which PfamProt2vec achieved accuracies of 1.0, protein members are more evolutionarily close to each other in corresponding phylogenetic trees.

The above results in **Figure 5.6**, **Table 5.8** and **Figure 5.7** demonstrated that PfamProt2vec performed worse in protein families with lower homologous similarities, and similar correlation was observed in the results of HHMER 3.1 as well.

**The number of families sharing members with each family.** According to the results shown in **Figure 5.6** (f), the prediction performance of PfamProt2vec was most significantly affected by the number of related families of each protein family. For a specific protein family PF, a higher value of related families indicates that protein sequences in PF are also annotated as family members of a higher variety of other protein families. For example, the protein EGFR\_HUMAN in **Figure 5.1** was annotated as the family members of four different protein families. This factor is also related to the factor, the average length of annotated sequences, that was demonstrated in **Figure 5.6** (c) because functional domains within the same protein sequence are more likely to be short.

By combining **Figure 5.6** (c) and (f), PfamProt2vec achieved less satisfying prediction performance in protein families with more related families and with shorter annotated sequences. This indicated the potential issues of predicting protein families at

---

<sup>33</sup> Please refer to [http://epidemic.bio.ed.ac.uk/how\\_to\\_read\\_a\\_phylogeny](http://epidemic.bio.ed.ac.uk/how_to_read_a_phylogeny) for more information about how to read a phylogenetic tree.

the whole sequence-level. It is only applicable to single-functional proteins, such as the *lipocalin* mentioned in **Section 5.2**.

## 5.7 Case study: Tyrosine kinases and Protein kinase domain

Among the 25 human protein families, we performed independent tests for *Tyrosine kinase* (PF07714) and *Protein kinase domain* (PF00069). Tyrosine kinase is an enzyme that can transfer a phosphate group from ATP to the amino acid tyrosine on the protein [70]. Protein kinase domain is a structurally conserved domain containing the catalytic function of protein kinases [70]. Although there exists an overlap between the two families, Protein kinase domain (PF00069) has a broader range including serine and threonine kinases.

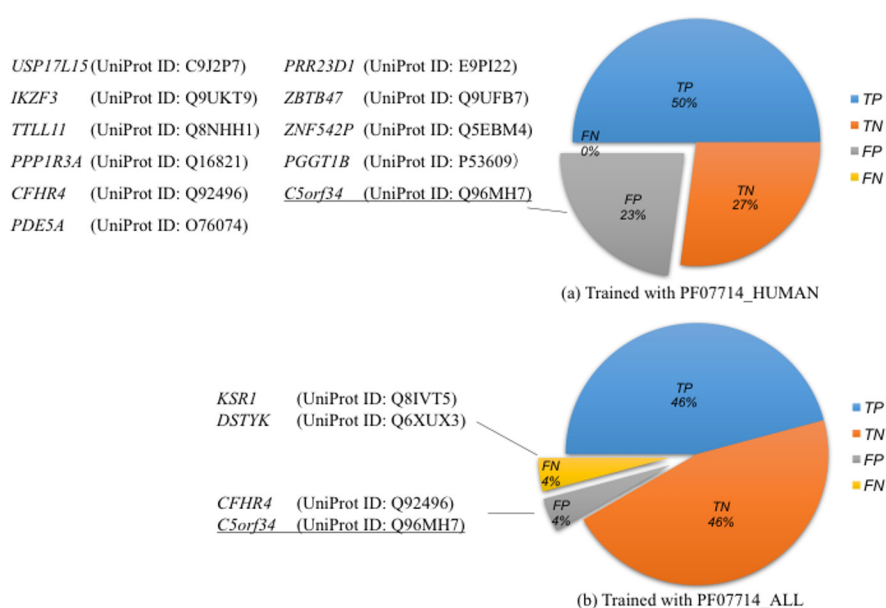
**Experimental design.** We prepared datasets based on the UniProt-Pfam dataset for cross-validation and independent test, respectively. Firstly, we extracted all human protein members and proteins members from all organisms for Tyrosine kinases (PF07714), denoted as PF07714\_HUMAN and PF07714\_ALL, respectively. We also extracted all human protein members and protein members from all organisms for Protein kinase domains (PF00069), denoted as PF00069\_HUMAN and PF00069\_ALL, respectively. We then divided each of the four sub datasets, *i.e.* PF07714\_HUMAN, PF07714\_ALL, PF00069\_HUMAN and PF00069\_ALL, into a training dataset and a validation dataset respectively from annotations created before and after Aug 23, 2008. Finally, negative sampling was performed for each dataset to ensure a ratio of 1:1 between the positive examples and the negative examples.

**Table 5.9** provides the statistics for the datasets used in the case study of *Tyrosine kinase* (PF07714) and *Protein kinase domain* (PF00069), respectively. For each of these two protein families, 10-fold cross-validation was performed on training datasets, and the model with the best prediction performance was used to apply to the independent test datasets.

**Table 5.9** Statistics of the constructed cross-validation and independent test datasets for protein family *Tyrosine kinase* (PF07714) and *Protein kinase family* (PF00069).

Dataset Name	Family Name	Organism	Cross-validation	Independent test
PF07714_HUMAN	Protein tyrosine kinase	Human	206	48
PF07714_ALL	Protein tyrosine kinase	All	805	48
PF00069_HUMAN	Protein kinase domain	Human	698	154
PF00069_ALL	Protein kinase domain	All	5130	154

**Result analysis.** Figure 5.8 shows the prediction performance of the family *Tyrosine kinase* (PF07714) in the independent test. In both cases, true positive (TP), true negative (TN), false positive (FP) and false negative (FN) predictions were reported in a pie graph, where the protein names of FP and FN predictions were listed. Using models trained only with human proteins, we successfully predicted 22 out of 24 positive examples and 22 out of 24 negative examples. The two proteins that were incorrectly predicted were *KSR1* (UniProt ID: Q8IVT5) and *DSTYK* (UniProt ID: Q6XUX3). As a comparison, these two proteins were correctly predicted as members of *Tyrosine kinase* (PF07714) by the models trained with proteins members of all organisms.



**Figure 5.8** Independent test in protein family *Tyrosine kinase* (PF07714), trained with human proteins and all proteins respectively.

To further investigate the prediction results, we calculated the prot2vec-based cosine similarities (see [Eq. 5.1]) between *KSR1* and each positive example in PF07714\_HUMAN and PF07714\_ALL, respectively. We found that *KSR1* was most similar to Ksr1 (UniProt ID: Q61097), a mouse protein, with a cosine similarity of 0.874, while the largest cosine similarity between *KSR1* and a human protein *KSR2* (UniProt ID: Q6VAB6) was 0.475. The difference in the cosine similarity explained the better performance of the models trained with PF07714\_ALL compared to that trained with PF07714\_HUMAN. It also validated our conclusion that protein members of all organisms should be combined to achieve a better prediction performance.

In terms of false positive predictions, we searched their conserved domain using HMMER [69] and CD-search [313], and none of them contained apparent protein kinase domains. However, some proteins do possess functional properties that are commonly shared by tyrosine kinases: 1) *Tyrosine kinases* are closely associated with cancer development [314], while *C5orf34* is over-expressed in liver cancer tissues according to The Human Protein Atlas [315]. It was also one of the 10 most differentially expressed proteins when comparing borderline and malignant phyllodes tumours [316]; 2) Many receptor tyrosine kinases auto-phosphorylate [317, 318], while *C5orf34* is also phosphorylated on Y528 in human [319]; 3) *Tyrosine kinases* regulate cell activities [320], while *USP17L15* regulates different cellular processes that may include cell proliferation, progression through the cell cycle, apoptosis, cell migration, and the cellular response to viral infection [39]; *IKZF3* plays an essential role in regulation of B-cell differentiation, proliferation and maturation to an effector state [39]; and *PPP1R3A* is essential for cell division [39]. Altogether, all these clues indicate that it is worthwhile to experimentally investigate the pathological roles of these false positive predictions and their potential involvement in kinase-regulated signalling pathways.

For *Protein kinase domain* (PF00069), we correctly predicted 63 out of 77 protein members and 74 out of 77 negative examples with the model trained on the dataset

PF00069\_HUMAN. The three false positive predictions were *EPHA2* (UniProt ID: P29317), *L3MBTL1* (UniProt ID: Q9Y468) and *PIP4K2C* (UniProt ID: Q8TBX8). According to the Swiss-Prot/UniProt databases, *EPHA2* is currently annotated as a member of *Tyrosine kinase* (PF07714), whose members should be annotated as members of *Protein kinase domain* (PF00069) as well. However, according to Pfam, *EPHA2* is not annotated as a member of Protein kinase domain. It indicates that the Pfam classification system is not perfectly complete. We also found that the protein *PIP4K2C* belongs to the family PIP5K (PF01504) and it is a special kinase that phosphorylates the D-5 position of the inositol ring of Phosphatidylinositol 4-phosphate [321].

## 5.8 The implementation of the online webserver

The different implementations of the generation of the distributed representation of protein sequence, prot2vec, and their application to protein family prediction is publicly available at <http://118.138.240.130> as an online service at for academic use.

The service of prot2vec generation returns three different types of prot2vecs, including  $\text{prot2vec}^{\text{inference}}$ ,  $\text{prot2vec}^{\text{add}}$  and  $\text{prot2vec}^{\text{non-overlap}}$ , which correspond to Fetch/Infer, Add and Stack in **Table 4.2**. It takes a single protein sequence or a list of protein sequences as input, either in the format of FASTA or raw amino acid sequences. After submission, protein sequences are split into overlapping or non-overlapping 3-gram biological words which can then be queried against a pre-trained distributed memory network. Generated prot2vecs are in the format of .txt (ASCII), downloadable as plain/text file.

The service of protein family prediction returns the protein families of submitted protein sequences or the protein members of selected protein families. In addition, benchmark service is provided, in which the predicted results are evaluated against the benchmark dataset UniProt-InterPro. The graph-based visualisation includes a bar chart

demonstrating true positive, true negative, false positive and the false negative predictions, a table with precision, sensitivity and specificity scores and a pie graph illustrating the overall performance summary.

## 5.9 Chapter summary

To validate the effectiveness of prot2vec in whole sequence-level predictions, in this chapter, we systematically analysed the performance of prot2vec in protein family prediction.

Firstly, we calculated the cosine distance between protein families in prot2vec-based vector space and visualised the prot2vecs of proteins in different protein families using PCA [265]. According to the vector space analysis, protein families with similar biological functions have smaller cosine distances in prot2vec-based vector space compared with those with dissimilar biological functions. The visualisation of prot2vecs with respect to different protein families demonstrated that the distributed representation of protein sequences captured the ‘semantics’, *i.e.* their biological functions. These observations provided the confidence of using prot2vec as the input feature for protein family prediction.

Based on the promising results of the vector space analysis, we applied prot2vec to SVM-based protein family prediction (PfamProt2vec). We first compared the prediction performance of PfamProt2vec with two different implementation strategies, *i.e.* the prot2vec<sup>inference</sup> that was proposed in this thesis and the prot2vec<sup>add</sup> (ProtVec) that was proposed in [193]. The results demonstrated that prot2vec performed better than prot2vec<sup>add</sup> for 888 out of 1,033 proteins and improved the average prediction performance by 13.7% and 8.9% in terms of sensitivity and precision, respectively. Considering the difference between prot2vec<sup>inference</sup> and prot2vec<sup>add</sup>, the superior performance of prot2vec<sup>inference</sup> indicates that the order of amino acids plays an important role in the generation of prot2vecs for protein sequence-level prediction tasks.

We further compared the prediction performance of *PfamProt2vec* to that of *SVM-Prot 2016* and *HMMER 3.1*, among which the first employed the purely data-driven representation *prot2vec<sup>inference</sup>* as the input feature while the latter two respectively incorporated physicochemical properties-based features and homology profile-based features. Both *SVM-Prot 2016* and *HMMER 3.1* outperformed *PfamProt2vec* in terms of sensitivity but performed less satisfactorily than *PfamProt2vec* in terms of precision. Combining all performance evaluation scores, *PfamProt2vec* achieved comparable performance with *SVM-Prot 2016* but less preferable performance than that of *HMMER 3.1*.

To investigate the factors that may affect the performance of *PfamProt2vec*, we performed correlation analysis to investigate the potential factors that may affect the prediction performance of *PfamProt2vec*. We found that protein members from multiple organisms should be combined together to achieve better performance and protein families with more members are more likely to be better predicted. We also found that the performance of *PfamProt2vec* was affected by the sequential and homological similarity within each protein family and strongly affected by the number of related protein families in each protein family. These results revealed the disadvantage of the methodology of predicting protein families at the whole sequence-level.

In this chapter, we obtained the following important conclusions. Firstly, purely data-driven representation does not work perfectly in protein analysis, even though it achieved improved performance in other areas such as natural language processing. Secondly, protein sequence data alone is not sufficient enough to represent evolutionary relations between proteins. Thirdly, the distributed representation of protein sequences should therefore be used in combination with existing biological heuristic-based representations for protein sequence-based predictions. With this conclusion, in **Section 7.4**, we combined the purely data-driven representation *context2vec* with existing residue-level representations that are generated from homology profiles, physicochemical properties and structural information for phosphorylation site prediction.



# 6 Protein IDP/IDR and SSP prediction

## 6.1 Introduction

In **Section 4.2** and **Section 4.3**, we proposed the DeepS2P framework for predicting protein structural and functional properties and the multitask deep learning frameworks for predicting two or more protein structural properties simultaneously. In this chapter, we evaluate the performance of the DeepS2P framework and multitask frameworks in protein structural prediction tasks. We first applied the DeepS2P framework to the prediction of IDP/IDRs and SSPs. Moreover, to demonstrate the positive effect of predicting multiple protein structural properties simultaneously, we applied the Multi-task framework and the Cross-stitch framework to the simultaneously predict IDP/IDRs and SSPs, with SSP prediction as the supporting task for IDP/IDR prediction.

## 6.2 Protein intrinsic disorder prediction based on the DeepS2P framework

Historically, protein function has been closely associated with a unique, well-defined three-dimensional structure. This paradigm has often been referred to as the sequence-structure-function relationship [36]. However, intrinsically disordered proteins (IDPs) and proteins with intrinsically disordered regions (IDRs) elude this understanding. Their conformational states are highly heterogeneous and experimentally difficult to characterize since they do not show stable electron densities in crystal structure analysis [322]. Spectroscopic methods such as NMR have proven useful for experimentally studying the structural propensities and dynamics of IDPs [323]. However, due to the expensive nature of these experiments, computational methods have been proposed to predict IDPs and IDRs directly from amino acid sequences [101, 153, 207, 211-213, 324].

In this section, we apply the proposed DeepS2P framework to IDP/IDR prediction. The resulting predictive model is referred to as *DeepS2P-D*. We performed hyperparameter tuning to determine the structure of DeepS2P-D and independent tests to compare the performance of DeepS2P-D to those of the existing shallow and deep architectures.

### 6.2.1 An overview of IDP/IDR prediction

The task of IDP/IDR prediction seeks to assign the ordered and disordered states to each target residue  $r_i$  in each protein sequence  $S$ . Therefore, the IDP/IDR prediction is usually modelled as a binary classification task, where each example is classified into two classes, *ordered* and *disordered*, which can be formalised as follows,

$$M: g\left(\text{concat}(f_{\text{Const}}(r_{i-w}), f_{\text{Const}}(r_i), f_{\text{Const}}(r_{i+w}))\right) \rightarrow a, b \quad [\text{Eq. 6.1}]$$

where  $a, b \in 0, 1$  and  $a + b = 1$ . Instead of using one output neuron producing a value of 0 or 1, here 1-of-K coding is used in the output layer, where the two classes *ordered* and *disordered* are modelled with two output neurons, each of which produces the value of 0 or 1, with the restriction that only one of them produces 1 while the other produces 0.

According to the DeepS2P framework (see **Section 4.2**), the input feature  $\text{concat}(f_{\text{Const}}(r_{i-w}), f_{\text{Const}}(r_i), f_{\text{Const}}(r_{i+w}))$  is obtained by concatenating the constructed feature vectors of the residues within the sliding window of size  $L = 2 * w + 1$ , where  $w$  indicates the number of upstream and downstream residues that are included in the sliding window. For feature construction, we calculated the position-specific scoring matrix (PSSM) [99] for each residue within the sliding window as the only source of information for prediction. Combined with the position and residue type, the constructed feature vector  $f_{\text{Const}}$  of each residue is composed of 23 real-values, where the first value represents the position of the residue in the protein sequence, the second value indicates

the residue type ranging from -10.0 to 10.0, and the remaining 21 values collected from the generated PSSM.

### 6.2.2 Experimental design

**Datasets.** The training dataset for IDP/IDR prediction was downloaded from the web server of SPINE-D [212]. It contains 4,229 protein sequences with 1,037,572 residues labelled as '+', '-' and 'x' indicating disordered, ordered and unclear states respectively. Altogether, there are 103,190 disordered residues and 933,382 ordered residues, resulting in a ratio of 1:9 between positive and negative examples. For performance evaluation, we randomly divided the 4,229 proteins into ten subsets for 10-fold cross-validation. For independent tests, we applied our IDP/IDR prediction models to targets reported in CASP experiments [292, 325]. Specifically, we applied the model with the best performance on the validation set to the 117 targets in CASP9 [270] and the 94 targets in CASP10 [271, 292]. Due to experimental noise affecting the recognition of short disordered regions, short sequence segments consisting of less than four consecutive residues of the same ordered/disordered types were eliminated from the test [270, 271]. This left us with 2,417 disordered residues and 23,658 ordered residues in the CASP9 dataset and 1,502 disordered residues and 22,688 ordered residues in the CASP10 dataset. In this study, we compared our results with those of the top-ranked models listed in CASP9 and CASP10 assessments for predicting IDP/IDRs.

**Performance evaluation.** The prediction performance of IDP/IDR prediction was evaluated using three measures: the balanced accuracy *BACC*, Matthew's coefficient of correlation *MCC* and the area under the ROC curve (AUC) (Please refer to **Appendix A** for a detailed description of the evaluation measurement equations). Additionally, to obtain a more detailed performance evaluation, we also recorded the number of true positives *TP*, false positives *FP*, true negatives *TN* and false negatives *FN*. Among the three measures, *BACC* and *MCC* depend on the predefined cut-off threshold of the prediction score for methods that generate probabilities instead of binary predictions.

According to previous results from CASP9 and CASP10 [270, 271], we used 0.5 as the cut-off threshold. For both measures a higher score indicates better performance. The third score, AUC, is a widely-used probability-based measure for which a score of 1 represents the perfect prediction and a score of 0.5 indicates a random prediction.

### 6.2.3 Determination of hyper-parameters

When applying the DeepS2P framework to IDP/IDR prediction, one of the crucial steps is to determine the hyper-parameters of the final models. According to **Table 4.5**, hyper-parameters are used to determine the model structure and guide the process of model training. In this section, we perform comparative experiments to select the best values for hyper-parameters including the number of epochs  $m$ , the mini batch size  $n$ , the drop-out rate  $\vartheta$ , the learning rate  $\partial$ , the number of hidden layers  $|H|$ , and the size of the contextual window sizes  $L$  in the input layer.

We set the basic parameter values as follows,  $m=1$ ,  $n=64$ ,  $\vartheta=0.75$ ,  $\partial=1e-5$ ,  $|H|=1$ , and  $L=9$ . For each of the hyper-parameters, we trained models with different parameter values while keeping the basic values of other hyper-parameters unchanged. **Table 6.1** demonstrates the results for hyper-parameter tuning for IDP/IDR prediction using the DeepS2P framework. The prediction performance of each model was evaluated in terms of the cross entropy, the accuracy (ACC) and their standard errors. Note that the cross entropy was used as the objective function in classification tasks, for which a smaller value indicates a better prediction performance.

**Table 6.1** Hyper-parameter tuning for DeepS2P-D in IDP/IDR prediction.

Epochs	Cross entropy	std. Entropy	ACC	std. ACC
1	0.245	0.022	0.915	0.010
3	0.221	0.017	0.926	0.007
5	<b>0.216</b>	<b>0.017</b>	<b>0.927</b>	<b>0.007</b>

Batch size	Cross entropy	std. Entropy	ACC	std. ACC
32	<b>0.234</b>	<b>0.018</b>	<b>0.918</b>	<b>0.009</b>
64	0.245	0.022	0.915	0.010
128	0.257	0.022	0.912	0.010
256	0.271	0.023	0.905	0.011

Win sizes	Cross entropy	std. Entropy	ACC	std. ACC
5	0.253	0.020	0.912	0.010
9	0.245	0.022	0.915	0.010
15	<b>0.241</b>	<b>0.019</b>	<b>0.915</b>	<b>0.010</b>
19	0.243	0.019	0.915	0.010

Drop-out rate	Cross entropy	std. Entropy	ACC	std. ACC
0.2	0.261	0.022	0.910	0.010
0.5	0.248	0.021	0.914	0.010
0.75	0.245	0.022	0.915	0.010
0.9	<b>0.242</b>	<b>0.019</b>	<b>0.915</b>	<b>0.010</b>

Hidden layers	Cross entropy	std. Entropy	ACC	std. ACC
1	0.245	0.022	0.915	0.010
2	0.232	0.018	0.920	0.009
3	<b>0.228</b>	<b>0.017</b>	<b>0.923</b>	<b>0.008</b>

Learning rate	Cross entropy	std. Entropy	ACC	std. ACC
0.001	<b>0.207</b>	0.017	<b>0.931</b>	<b>0.007</b>
0.0001	0.213	<b>0.016</b>	0.929	0.007
0.00001	0.245	0.022	0.915	0.010

\* The best performance in terms of different evaluation measurements was highlighted as bold.

According to the results in **Table 6.1**, some hyper-parameters have more impact on the prediction performance, such as the number of epochs, the size of mini-batches, the number of hidden layers and the learning rate. In terms of the size of mini-batches, the best performance was achieved with the batch size 32. The average accuracy was improved to 0.918 from 0.905 that was achieved with batch size 256. In terms of the number of hidden layers, the average accuracy was improved from 0.915 to 0.923 when the hidden layers were improved from 1 to 3. It indicates that the addition of more hidden layers played a positive effect on the performance in IDP/IDR prediction. In terms of different learning rates, the best accuracy score 0.931 was achieved with a learning rate of 0.001. Among all the three tested learning rates, better performance was achieved with large learning rate.

As for the number of epochs, the best accuracy score 0.927 was achieved with 5 epochs of training. In practice, we employ an auto-stop policy to dynamically determine the stop of the training process. The stop condition was set so that the training process for each fold would terminate only when the cross entropy on the valid dataset had not improved for 200 epochs since the last improvement. According to the results in **Table 6.1**, we selected the values for respective hyper-parameter as follows,  $m=N/A$ ,  $n=32$ ,  $\vartheta=0.9$ ,  $\partial=1e-3$ ,  $|H|=3$ , and  $L=15$ .

#### **6.2.4 Comparison between deep architectures and shallow architectures**

The performances of DeepS2P-D in terms of TP, FP, TN, FN, BACC, MCC and AUC were evaluated on CASP9 and CASP10 targets [270, 271]. The performance of DeepS2P-D was compared to that of the top four methods listed in the IDP/IDR prediction assessment in CASP9 and CASP10 [270, 271], respectively. We also compared the prediction of DeepS2P-D to that of the more recent DeepCNF-D model, which used the deep CNF (see **Section 2.1.2**) to capture local correlations among the input neighbouring residues and independencies of the predicted ordered/disordered labels among adjacent residues [158].

The four models with the best performance in CASP9 assessment are PrDOS2 [211], DisoPred3C [102], MultiCom [326] and SPINE-D [212]; the four models in CASP10 assessment are PrDos-CNF [213], DISOPRED3 [101], Biomine-dr-mixed and Biomine-dr-pdb-c [327]. The PrDOS2 model combines sequential and template information using an SVM-based model. The PrDos-CNF model employs deep convolutional neural fields to capture correlations between the predicted labels of neighbouring residues. DisoPred3C and DISOPRED3 [101] are also SVM-based and trained on high-resolution X-ray structures. The MultiCom model and the SPINE-D model combine multiple sources of information including amino acid-based features, PSSMs, predicted secondary structures, and solvent accessibilities [212]. The MultiCom model is based on one-dimensional recurrent neural networks, and the SPINE-D model is based on two-layer neural networks. Finally, the Biomine-dr-mixed and Biomine-dr-pdb-c models are both SVM-based and combine multiple sources of predicted information.

The deep learning model DeepS2P-D introduced in this section is derived from the singletask deep learning framework DeepS2P, using PSSMs as the input feature. Among all nine compared models, three were based on deep neural networks including PrDos-CNF, DeepCNF-D and SPINE-D, five were SVM-based models, and one used recurrent neural networks.

**Table 6.2** Performance comparison for IDP/IDR prediction applied to CASP9 proteins.

Predictors	Targets	TP	FP	TN	FN	BACC	MCC	AUC
<i>Shallow models</i>								
PrDOS2	117	<b>1,468</b>	2,340	21,318	<b>949</b>	<b>0.754</b>	0.418	0.855
DisoPred3C	117	839	<b>180</b>	<b>23,478</b>	1,578	0.670	0.508	0.854
MultiCom	110	953	934	21,695	1,310	0.690	0.413	0.853
<i>Deep models</i>								
SPINE-D	117	1,399	2,774	20,884	1,018	0.731	0.365	0.832
DeepCNF-D	117	-	-	-	-	0.752	0.486	0.855
DeepS2P-D	117	903	256	23,402	1,514	0.681	<b>0.510</b>	<b>0.856</b>

\* The best performance in term of TP, FP, TN, FN, BACC, MCC and AUC was highlighted as bold.



**Table 6.3** Performance comparison for IDP/IDR prediction applied to CASP10 proteins.

Predictors	Targets	TP	FP	TN	FN	BACC	MCC	AUC
<i>Shallow models</i>								
DISOPRED3	94	607	201	22,487	895	0.698	0.531	0.897
Biomine-dr-mixed	94	628	368	22,320	874	0.701	0.488	0.890
Biomine-dr-pdb-c	94	579	290	22,398	923	0.686	0.483	0.886
<i>Deep models</i>								
PrDos-CNF	94	<b>657</b>	287	22,401	<b>845</b>	0.712	0.529	<b>0.907</b>
DeepCNF-D	94	-	-	-	-	<b>0.764</b>	0.474	0.898
DeepS2P-D	94	629	<b>176</b>	<b>22,512</b>	873	0.706	<b>0.533</b>	0.895

\* The best performance in term of TP, FP, TN, FN, BACC, MCC and AUC was highlighted as bold.

According to the evaluation on CASP9 in **Table 6.2**, DeepS2P-D achieved the best prediction performance among all compared IDP/IDR prediction models. In terms of BACC, PrDOS2 still performed best among all methods with a score of 0.754. In comparison, the DeepS2P-D model introduced in this section achieved BACC scores of 0.660. As for MCC, DeepS2P-D achieved the best performance, improving the MCC scores by 2.4% and 14.5%, respectively, when compared with the MCC scores of the DeepCNF-D and SPINE-D model. According to the prediction statistics based on TP, FP, TN and FN, PrDOS2 achieved the best sensitivity by correctly predicting 1,468 of 2,417 positive examples, while DisoPred3C achieved the best precision by correctly predicting 839 positive examples in 1,019 positive predictions. The different results of the two models indicate that PrDOS2 makes "riskier" predictions to improve sensitivity, while DisoPred3C tends to make more "cautious" predictions to ensure high precision. In comparison, the performance of DeepS2P-D model represents a trade-off of the prediction performance of the above two models, with a tendency to make more positive but cautious predictions. Specifically, it correctly predicted more positive examples than DisoPred3 and, at the same time, obtained an FP to TP ratio below 1:3.

When applied to CASP10 proteins, we observed superior prediction performance achieved by deep learning models compared with that achieved by shallow models. With the most true-positive predictions, PrDos-CNF achieved the best AUC score of 0.907. Between the other two deep learning models, DeepCNF-D achieved the best BACC score of 0.764 and DeeSP2-D achieved the best MCC score of 0.533. In terms of the statistics of TP, FP, TN, and FN, PrDos-CNF achieved the best sensitivity by making 657 true positive predictions and 845 false negative predictions. In comparison, DeepS2P-D had the best precision by making 629 true positive predictions and 176 false positive prediction. This indicates that PrDos-CNF is more like to make positive predictions, but with a lower precision; while DeepS2P-D tends to make positive predictions that are more likely to be correct.

### 6.2.5 Conclusion

In this section, we applied the DeepS2P framework to the prediction of intrinsically disordered proteins and regions. The results in **Table 6.1** demonstrated that the addition of hidden layers played a positive effect in improving the prediction performance. It indicated the advantage of using deep learning model for IDP/IDR prediction over show non-linear models. This observation is further emphasized by the results in the independent tests conducted on the CASP9 and CASP10 targets. The DeepS2P-D model achieved comparable or superior performance compared to existing deep learning-based methods such as DeespCNF-D and PrDoc-CNF, and achieved superior performance than other methods including PrDOS2 and SPINE-D. The DeepS2P-D model represents a successful application of the proposed DeepS2P framework in predicting protein structural properties.

## 6.3 Protein SSP prediction based on the DeepS2P framework

### 6.3.1 An overview of SSP prediction

The task of protein SSP prediction seeks to assign the populations of the helix, strand and coil structures to each target residue  $r_i$  in each protein sequence  $S$ . The populations of each of the three secondary structure elements are represented as three real-valued numbers ranging from 0 to 1. When modelled using machine learning approaches, the task of protein SSP prediction can be treated as a regression task and represented as follows,

$$M: g(\text{concat}(f_{\text{const}}(r_{i-w}), f_{\text{const}}(r_i), f_{\text{const}}(r_{i+w}))) \rightarrow a, b, c \quad [\text{Eq. 6.2}]$$

where  $a, b, c \in (0, 1)$  and  $a + b + c = 1$ . The three real-valued outputs of the prediction task,  $a$ ,  $b$  and  $c$ , respectively represent the predicted population of helix, strands and coil. As for feature construction, we used the same feature vector that is introduced in the

prediction of IDP/IDR. More specifically, for each residue within the sliding window of size  $L = 2 * w + 1$ , a feature vector  $f_{const}$  of size 23 was constructed.

We refer to the deep learning model that is derived from the DeepS2P framework for the prediction of SSP as the *DeepS2P-P* model. In the rest of this section, we determine the structure of DeepS2P-P via hyper-parameter tuning, evaluate the performance of DeepS2P-P by comparing its prediction performance to existing methods for SSP prediction on constructed benchmark dataset, and applied the DeepS2P-P model for predicting SSPs for intrinsically disordered proteins and regions.

### 6.3.2 The difference between protein SSP prediction and SS prediction

The prediction of protein SSP should be distinguished from the prediction of protein SS. As introduced in **Section 1.2.3**, protein secondary structure refers to the intermediate form of protein local segments before they fold into the tertiary structure. It is described using the 3-class system where each residue is mapped to one of the three secondary structure elements or using the 8-class system where each residue is mapped to one of the 8 secondary structure elements. A number of computational methods and bioinformatics tools have been developed to assign secondary structures to residues in protein sequences using neural networks, support vector machines, hidden Markov models and deep learning models.

The prediction of SSP is different from the prediction of SS theoretically and practically. In SSP prediction, each residue in protein sequences is assigned to three real-valued populations indicating the electron densities of the 3 secondary structure elements, helix, strand and coil, for each residue. Theoretically, the dynamic SSP assignments are consistent with the discovery of IDP/IDRs, which represents a group of proteins that lack stable tertiary structures. However, the fixed SS assignments are consistent with the traditional protein structure paradigm, where protein functions are determined by fixed protein structures. Practically, the training of the models for SSP prediction is based on training examples that are labelled with three numeric real-values

$a$ ,  $b$ , and  $c$  where  $a, b, c \in (0, 1)$ , while the training of models for SS prediction is based on training examples that are labeled with binary categorical value 0 or 1.

Since the prediction of SSP and SS are theoretically and practically different from each other, it is unfair to compare the prediction methods of SSP prediction to those of the SS prediction, to evaluate the prediction of SSP prediction on existing benchmark datasets for SS prediction, or to evaluate the prediction of SSP using the same evaluation measurements with those of the SS prediction.

### 6.3.3 Experimental design

**Datasets.** The training dataset of protein SSPs was obtained from the original study of the  $\delta$ 2D method, a software that calculates SSPs from a protein's NMR chemical shifts. For detailed information, please refer to the s2D method [194]. There are 2,671 proteins with 362,702 residues in the  $\delta$ 2D dataset, among which 2,223 proteins were obtained from the BMRB database [93], with the remaining 448 proteins obtained from the RCSB PDB database in form of X-ray structures. Following the s2D method, proteins in the dataset were clustered into ten subsets with a 25% sequence identity cutoff. These subsets were then used for 10-fold cross-validation, in which, for each iteration, nine subsets were used for the purpose of model training and the remaining subset was used as the validation set. Again, following the s2D method, the prediction model was trained on proteins from both the BMRB database and the RCSB PDB database but validated only on proteins from the BMRB database.

**A novel benchmark dataset for SSP prediction.** In order to properly evaluate the performance of methods for SSP prediction in independent test, we constructed a novel benchmark dataset from the latest release of the BMRB database.

We downloaded the chemical shift annotations for 12,018 entries from the BMRB database (March 2018) [93] and created a candidate list of 2,293 BMR entries using the following filtering procedure. Firstly, we kept entries that have the same experimental conditions described in the s2D method [194]. More specifically, any entry that is not

sampled with pH between 5.5 and 8 or with temperature between 10 and 42 °C was removed from the candidate list. Secondly, we only kept the entries that have the sample type labelled as 'solution' and removed any entry with amino acid 'X'. Thirdly, we extracted the annotated values for the six backbone chemical shifts, e.g. CA, CB, CO(C), N, HA, and HN, and removed entries that are lack of at least one of the six backbone chemical shifts. Finally, we removed entries that appear in the s2D training datasets.

To obtain the SSP annotations for each candidate entry, we used the d2D method to generate the populations for helix, strands and coils according to their chemical shifts. To obtain the feature values for each candidate entry, we ran PSI-BLAST to generate the PSSMs according to their amino acid sequences. For 1,009 of the 2,293 entries, PSI-BLAST failed to find a matching hit. Therefore, we end up with a dataset of 1,284 BMR entries.

The resulting benchmark dataset BMR\_2018 is publicly available in the GitHub repository: <https://github.com/yxu132/multitask-ssp-diso>.

**Performance evaluation.** For the purpose of performance evaluation, we conducted both cross-validation and independent tests. In cross-validation tests, we split the training dataset into 10 folds, from which 9 folds were used for training and the remaining one folds was used for validation. This process of training-and-validation was repeated 10 times so that each example in the training dataset is used for both training and validation at least once. With cross-validation, we determined the hyper-parameters of the model DeepS2P-P and compared the performance of DeepS2P-P to that of the s2D method. In independent tests, we applied the best performing model among the 10 models trained during the process of cross-validation and applied it to the constructed benchmark dataset BMR\_2018. With the independent test, we compared the prediction performance of DeepS2P-P to that of the s2D method.

Following the methodology introduced in the s2D method, the prediction performance of the DeepS2P-P model in cross-validation tests was evaluated using three measurement scores including the *MSE* [Eq. 4.14] and the *mean absolute error (MAE)*

[Eq. 6.3] used to measure the difference between true values and predicted values during the process of training, and the *Pearson's coefficient of correlation* (PCC or  $R$ ) (see **Appendix A**) that is used to evaluate the linear correlation between true populations and prediction populations.

$$MAE(\hat{\mathbf{y}}, \mathbf{y}) = \frac{1}{n} \sum_{i=1}^n |\hat{y}_i - y_i| \quad [\text{Eq. 6.3}]$$

where  $\hat{\mathbf{y}}$  is the predicted results of the true label  $\mathbf{y}$ .

The three measurements MSE, MAE and  $R$  are calculated for the three secondary structure elements helix (H), strands (E) and coils (C), which are denoted as  $(MSE_H, MAE_H, R_H)$ ,  $(MSE_E, MAE_E, R_E)$  and  $(MSE_C, MAE_C, R_C)$ , respectively.

To analyse the correlation between the predicted SSPs and protein IDP/IDRs, we applied the trained DeepS2P-P model to intrinsically disordered proteins and regions.

#### 6.3.4 Determination of hyper-parameters

When the DeepS2P framework is applied to SSP prediction, the output layer  $O$  is composed of three real-valued neurons, for which the sigmoid function [Eq. 4.12] is used as the activation function, and the MSE score [Eq. 4.14] is used as the objective function. At the same time, in order to finalise the deep model structure specifically for the task of SSP prediction, we conducted comprehensive comparison experiments to tune the hyper-parameters that are mentioned in **Section 4.2.5**.

For each set of hyper-parameter settings, 10-fold cross-validation was conducted, where the deep models were trained on the 9 folds of the data, and the  $R_H$ ,  $R_E$ , and  $R_C$  scores were calculated on the remaining fold. **Table 6.4** demonstrates the prediction performance of corresponding models under different settings of hyper-parameters. Here, we reported the results for three hyper-parameters that demonstrated the most significant improvements, including the number of hidden layer  $|H|$ , the number of neurons in the input layer  $L$ , and the number of training epochs  $m$ .

**Table 6.4** The selection of the hyper-parameters for the single-task deep learning framework.

Hyper-parameters			Prediction performance		
$ H $	$L$	$m$	$R_H$	$R_E$	$R_C$
2	11	1	0.729±0.012	0.631±0.012	0.653±0.012
2	15	1	0.736±0.015	0.632±0.012	0.661±0.012
2	19	1	<b>0.741±0.018</b>	<b>0.638±0.023</b>	<b>0.662±0.023</b>
2	23	1	0.740±0.011	0.629±0.020	0.659±0.020
1	19	1	0.646±0.026	0.499±0.037	0.620±0.037
2	19	1	0.741±0.018	0.638±0.023	0.662±0.023
3	19	1	<b>0.763±0.011</b>	<b>0.672±0.018</b>	<b>0.677±0.018</b>
3	19	3	0.827±0.006	0.783±0.009	0.724±0.009
3	19	6	0.839±0.005	0.794±0.009	0.732±0.009
3	19	9	0.845±0.005	0.801±0.008	0.738±0.008
3	19	20	<b>0.852±0.003</b>	<b>0.806±0.008</b>	<b>0.742±0.015</b>

\* A significance level of 95% was used.

\* The best performance in terms of  $R_H$ ,  $R_E$ , and  $R_C$  was highlighted as bold.



### 6.3.5 Comparing linear models, non-linear shallow models and deep models

Characterizing IDP/IDRs using predicted protein SSPs provided a new avenue for quantitatively analysing IDPs/IDRs [194]. The s2D method achieved the state-of-the-art performance by modelling local sequential correlations and incorporating the predicted mean SSP of the entire protein sequence.

The DeepS2P-P model is based on the aforementioned single-task deep learning framework DeepS2P. Compared with the s2D method that uses three single-layered neural networks, DeepS2P-P employs neural networks with three convolutional hidden layers. Therefore, it represents a more expressive model which is capable of modelling more complex relations between the input features and output populations. At the same time, due to the design of the convolutional layers, the local sequential correlations among neighbouring residues are modelled layer-by-layer in a more consistent manner.

For comparison purposes, we introduce the linear regression model as the benchmark for predicting protein SSPs. Linear regression (LR) is a generic linear approach for modelling the relationship between a scalar-dependent variable and one or more relevant variables. In contrast, the s2D method represents non-linear shallow models, and the DeepS2P-P method represents non-linear deep learning models. A comparison of the three approaches illustrates how increasing model complexity benefits the prediction performance.

**Table 6.5** shows the performance of s2D, LR and DeepS2P-P for predicting protein SSPs *Helix (H)*, *Strand (E)* and *Coil (C)* on the  $\delta$ 2D validation set, with a significance level of 95%. According to these results, DeepS2P-P improved the Pearson correlation coefficient for helix, strands and coils by 0.035, 0.036 and 0.032, respectively, compared to s2D predictions. The LR model performed significantly worse than the other two methods, indicating that SSP prediction cannot be modelled by a linear approach.

**Table 6.5** Prediction performance comparison between s2D, LR, and DeepS2P-P for protein SSP prediction on the  $\delta$ 2D validation set.

	s2D	LR	DeepS2P-P
R(H)	0.817 $\pm$ 0.014	0.337 $\pm$ 0.052	<b>0.852<math>\pm</math>0.008</b>
R(E)	0.770 $\pm$ 0.039	0.379 $\pm$ 0.047	<b>0.806<math>\pm</math>0.014</b>
R(C)	0.710 $\pm$ 0.039	0.431 $\pm$ 0.047	<b>0.742<math>\pm</math>0.027</b>
MSE(H)	0.038 $\pm$ 0.004	0.107 $\pm$ 0.006	<b>0.032<math>\pm</math>0.004</b>
MSE(E)	0.024 $\pm$ 0.002	0.051 $\pm$ 0.004	<b>0.022<math>\pm</math>0.002</b>
MSE(C)	0.041 $\pm$ 0.002	0.067 $\pm$ 0.004	<b>0.036<math>\pm</math>0.002</b>
MAE(H)	0.140 $\pm$ 0.008	0.278 $\pm$ 0.008	<b>0.119<math>\pm</math>0.008</b>
MAE(E)	0.113 $\pm$ 0.006	0.190 $\pm$ 0.008	<b>0.101<math>\pm</math>0.008</b>
MAE(C)	0.158 $\pm$ 0.006	0.218 $\pm$ 0.009	<b>0.143<math>\pm</math>0.006</b>

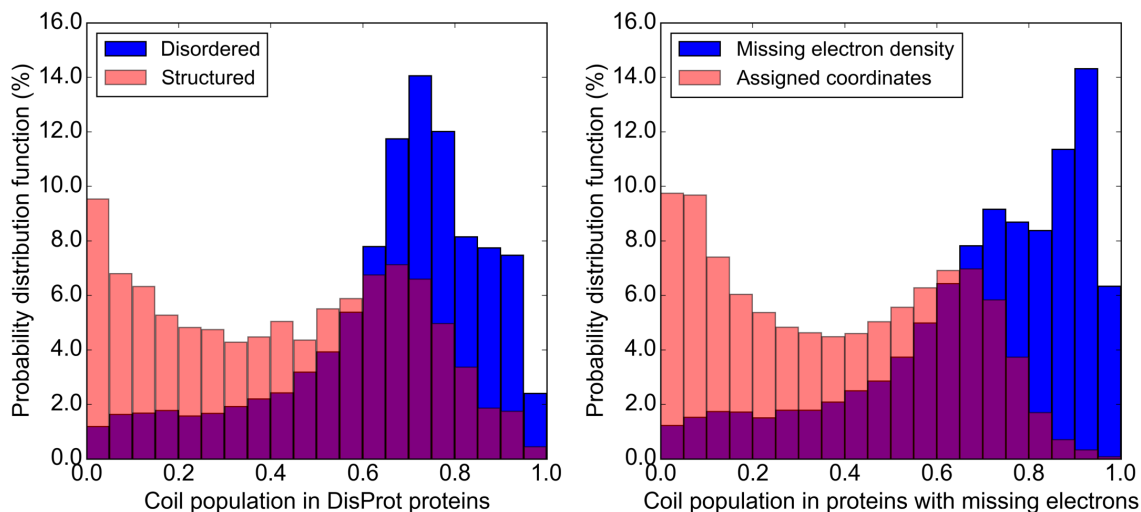
\* The best performance in terms of *R*, MSE, and MAE was highlighted as bold.

Here, LR, s2D and DeepS2P-D represent linear models, non-linear shallow models and deep models, respectively. They are train on exactly the same input data and the same set of selected features (including the indexes of the amino acids, amino acid types, and homology profiles generated using PSI-BLAST). In this way, the size of the training dataset becomes a controlled variable and the training procedure will not unfairly favour the deep learning model. Therefore, the significantly superior performance of DeepS2P-D was purely due to the deep learning architecture itself.

### 6.3.6 Validation of DeepS2P-P on disordered proteins and regions

For the 694 proteins reported in the old DisProt database [89] and the 1,304 chains reported to have missing electron density [328], we applied DeepS2P-P and observed the same structural characterizations for residues that are ordered and disordered and residues that have assigned coordinates and missing electron density, respectively. According to the left histogram in **Figure 6.1**, disordered regions are more likely to have higher coil populations (80.7% distributed between 0.5 and 1.0), while structured regions have lower coil populations. Similar to **Figure 6.1** (b) reported in the s2D method [194],

the number of structured residues with coil populations reached a peak value at 0.6, which indicates that some structured regions may be found in loops.



**Figure 6.1** Protein secondary structure populations of structured and unstructured proteins and regions.

According to our results, another peak value is around the coil population of 0.0. This is consistent with our expectation that structured regions are more likely to be found in helixes and strands, which was not demonstrated in **Figure 6.2 (b)** of the s2D method. Similarly, in the right histogram in **Figure 6.1**, the coil populations of 81.3% of the regions with missing electron density are distributed between 0.5 and 1.0, while those of regions with assigned coordinates are mainly distributed between 0 and 0.7. Compared to the s2D methods, DeepS2P-P predicted more residues with assigned coordinates to have low coil populations below 0 and 1. This indicates the superior performance of DeepS2P-P compared to that of the s2D method.

### 6.3.7 Validation of DeepS2P-P on benchmark datasets

To validate the performance of DeepS2P-P and the s2D method, we further applied these two predictive models to our constructed benchmark dataset BMR\_2018. Please refer to **Section 6.3.3** for detailed description of the construction procedure for this

benchmark dataset. **Table 6.6** demonstrates the prediction performance for DeepS2P-P and the s2D method for protein SSP prediction on BMR\_2018.

**Table 6.6** Performance comparison between DeepS2P-P and the s2D method on the independent benchmark BMR\_2018.

	DeepS2P-P			The s2D method		
	<i>Helix</i>	<i>Strands</i>	<i>Coils</i>	<i>Helix</i>	<i>Strands</i>	<i>Coils</i>
<i>R</i>	<b>0.832</b>	<b>0.796</b>	<b>0.652</b>	0.798	0.761	0.630
<i>MSE</i>	<b>0.040</b>	<b>0.029</b>	<b>0.048</b>	0.047	0.033	0.049
<i>MAE</i>	<b>0.129</b>	<b>0.119</b>	<b>0.171</b>	0.153	0.132	0.175

\* The best performance in terms of *R*, *MSE*, and *MAE* was highlighted as bold.

According to the results shown above, the Deep2DP-P method has improved the respective Pearson's coefficient of correlations by 4.1%, 4.4% and 4.9% for helix, strand and coil populations compared to those achieved by the s2D method. Correspondingly, DeepS2P-P decreased the MSE and MAE for helix, strand and coil populations as well. Its performance represents the state-of-the-art performance for this particular prediction task at the moment.

### 6.3.8 Conclusion

In this section, we applied the DeepS2P framework to protein SSP prediction and achieved superior performance in comparison with the existing s2D method in both cross-validation tests and independent tests. We also applied the trained DeepS2P-P model to disordered proteins in the DisProt database and to the regions from the RCSB PDB database that are observed with missing electron density, from which we observed the qualitative correlation between protein secondary structure populations and protein intrinsic disorder. More specifically, protein regions with higher coil populations are more likely to be in a disordered state. This observation inspired the application of multitask

deep learning frameworks for predicting two or more protein structural properties that is introduced in **Section 6.4**.

In this application, we also constructed a novel benchmark dataset for protein SSP prediction. To the best of our knowledge, it represents the first benchmark dataset for this particular prediction task. We envisage that it will be useful for independent test and benchmarking for SSP prediction in the future.

## **6.4 Simultaneous prediction of IDP/IDR and SSP using multitask frameworks**

Recently, the  $\delta$ 2D method was developed for calculating the probability distribution of SSPs of IDPs/IDRs from their NMR chemical shifts [66], based on which the s2D method was developed to predict SSPs of IDPs/IDRs directly from their amino acid sequences [194]. This allowed quantitative characterizing of IDP/IDRs in terms of their structural properties for the first time. According to reported validation results, the s2D method suggests that disordered states populate random coils preferably and have lower helix and strand populations.

In principle, better prediction performance of protein SSPs and IDPs/IDRs can be obtained by predicting SSPs and IDPs/IDRs simultaneously within a multitask framework. More specifically, because of the correlation between the population of coil structures and their ordered/disordered states, the representation learned for predicting SSPs is expected to be partially useful for the prediction of the unstructured proteins, and vice versa. The most recent deep learning-based methods including DeepCNF-D [158], AUCpreD [207] and SPOT-disorder [153] which use protein secondary structure annotation as part of the features for predicting unstructured proteins. However, it is not accurate to assign fixed secondary structure elements for predicting regions that do not form stable states. Given both SSPs and IDPs/IDRs are unknown, we propose using a multitask deep learning framework to simultaneously predict both.

In this section, we applied the Multi-task framework and the Cross-stitch framework introduced in **Section 4.3**, for predicting SSPs and IDPs/IDRs simultaneously, with both hard parameter sharing and cross-stitch-based soft parameter sharing; namely, Multi-task-D and Cross-stitch-D. We compared the prediction performance of DeepS2P-D, Multi-task-D, Cross-stitch-D and several state-of-the-art IDP/IDR prediction methods, including PrDOS2 [211], DisoPred3C [102], MultiCom [326], SPINE-D [212], Biomine [327], Prdos-CNF [211], DISOPRED3 [101] and DeepCNF-D [158], and observed improved performance when protein SSP prediction was accounted for as the second task in a multitask framework. For Cross-stitch-D, we generated cross-stitch units between SSP and IDP/IDR prediction, providing a quantitative view of the correlation between these two tasks. Furthermore, we investigated the different predictions obtained from DeepS2P-D and Multi-task-D to demonstrate the positive effect of the multitask design.

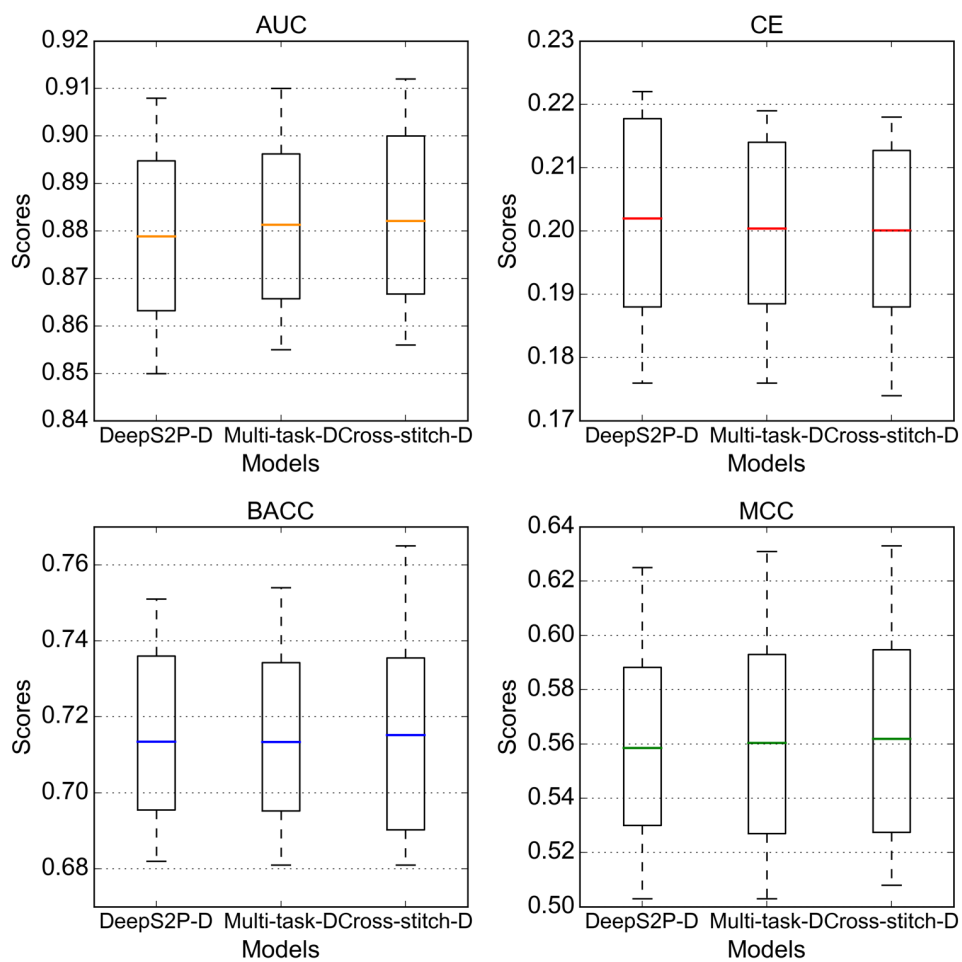
#### **6.4.1 Experimental design**

The datasets for IDP/IDR prediction and SSP prediction are the same as those introduced in **Section 6.2.2** and **Section 6.3.3**. Similarly, the same evaluation measurements were used for IDP/IDR and SSP prediction respectively (see **Section 6.2.2** and **Section 6.3.3**). It is noteworthy that both the Multi-task-D and the Cross-stitch-D models are pre-trained using the alternate training strategy that was introduced in **Section 4.3.4**. Here, the SSP prediction was used as the supporting task for IDP/IDR prediction. Therefore, during the process of alternate training, corresponding models were only updated and saved when the performance of IDP/IDR prediction was improved on the SPINE-D validation dataset.

#### **6.4.2 Comparison between single-task models and multitask models**

To optimise the parameters for IDP/IDR prediction, we updated models only when their performance improved on the SPINE-D validation set during training. We applied this

strategy to both the hard parameter sharing Multi-task framework and the cross-stitch multitask framework, resulting in the Multi-task-D model and the Cross-stitch-D model. We compared the prediction performance of DeepS2P-D, Multi-task-D and Cross-stitch-D by performing 10-fold cross-validation on the 4,299 protein targets in the SPINE-D dataset. **Figure 6.2** provides an overview of the performance comparison in terms of AUC, CE, BACC and MCC. In each sub-figure, the average scores are labelled with coloured lines and a box plot extended to the lower to the upper quartile values is demonstrated for each of the three compared models.



**Figure 6.2** Prediction performance of deep learning models for IDP/IDR prediction using 10-fold cross-validation.



While all hyper-parameters remained the same, the Multi-task-D model improved the average AUC score by 0.002 and decreased the cross-entropy loss by 0.002 when compared with DeepS2P-D, while the Cross-stitch-D model improved the average AUC score by 0.003 and decreased the cross-entropy loss by 0.002 when compared with DeepS2P-D. These improvements demonstrated the positive effect of combining the task of SSP prediction (as a supporting task) with that of IDP/IDR prediction. Compared to the single-task DeepS2P-D model, the Multi-task-D and Cross-stitch-D model have the advantage of simultaneously predicting SSPs and IDPs/IDRs, thereby allowing the incorporation of representations learned for the other task as the additional supportive evidence and the global optimisation of all parameters of both tasks.

#### **6.4.3 Independent test on CASP 9 and CASP 10 targets**

Similar to **Section 6.2.4**, the performance of DeepS2P-D, Multi-task-D and Cross-stitch-D in terms of TP, FP, TN, FN, BACC, MCC and AUC were evaluated on the CASP9 targets. The performance was compared with the top four models listed in the IDP/IDR prediction assessment in CASP9 and CASP10, respectively, as well the more recent DeepCNF-D model [158]. Feature groups and algorithms of compared models are listed in **Table 2.1**. All three models DeepS2P-D, Multi-task-D and Cross-stitch-D introduced in this study are based only on the homology profile PSSM, derived from the single-task framework DeepS2P or from the multitask deep learning frameworks.

**Table 6.7** Performance comparison for IDP/IDR prediction (CASP9).

	<b>Targets</b>	<b>TP</b>	<b>FP</b>	<b>TN</b>	<b>FN</b>	<b>BACC</b>	<b>MCC</b>	<b>AUC</b>
PrDOS2	117	<b>1,468</b>	2,340	21,318	<b>949</b>	<b>0.754</b>	0.418	0.855
DisoPred3C	117	839	<b>180</b>	<b>23,478</b>	1,578	0.670	0.508	0.854
MultiCom	110	953	934	21,695	1,310	0.690	0.413	0.853
SPINE-D	117	1,399	2,774	20,884	1,018	0.731	0.365	0.832
DeepCNF-D	117	-	-	-	-	0.752	0.486	0.855
DeepS2P-D	117	903	256	23,402	1,514	0.681	0.510	0.856
Cross-stitch-D	117	883	248	23,410	1,534	0.677	0.505	0.860
Multi-task-D	117	905	244	23,414	1,512	0.682	<b>0.514</b>	<b>0.867</b>

\* The best performance in terms of TP, FP, TN, FN, BACC, MACC, and AUC was highlighted as bold.

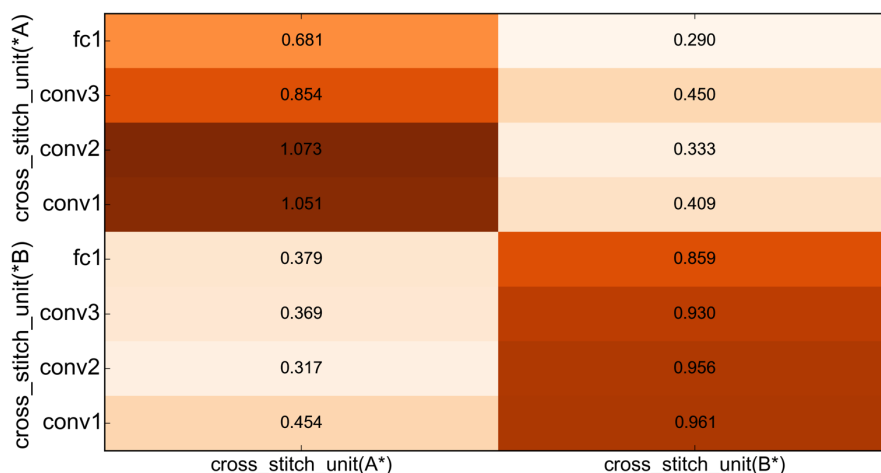
According to the evaluation on CASP9 in **Table 6.7**, compared with the other methods, Cross-stitch-D and Multi-task-D improved the AUC score to 0.860 and 0.867, respectively. In terms of BACC, PrDOS2 still performed best among all models with a score of 0.754. In comparison, the DeepS2P-D, Cross-stitch-D and Multi-task-D achieved BACC scores of 0.660, 0.677 and 0.682, respectively. As for MCC, Multi-task-D achieved the best, improving the MCC score by 2.8%, as compared to the MCC score of the DeepCNF-D model. According to the prediction statistics based on TP, FP, TN and FN, Multi-task-D achieved superior performance to that of DeepS2P-D by making more true-positive and true-negative prediction and less false-positive and false-negative predictions.

#### 6.4.4 Quantitative correlation between IDP/IDR and SSP predictions

Besides simultaneously predicting IDPs/IDRs and SSPs, the Cross-stitch-D model also automatically learned the linear correlation between each of the corresponding layers for these two tasks. The cross-stitch unit  $M$  in the Cross-Stitch framework is defined as follows (see **Section 4.3.3**),

$$M = \begin{bmatrix} \alpha_{AA} & \alpha_{BA} \\ \alpha_{AB} & \alpha_{BB} \end{bmatrix}$$

It was populated with real-valued correlations for convolutional layers conv1, conv2 and conv3 and the fully-connected layer fc1. This is illustrated as a heat map in **Figure 6.3**, where darker orange indicates strong dependence.



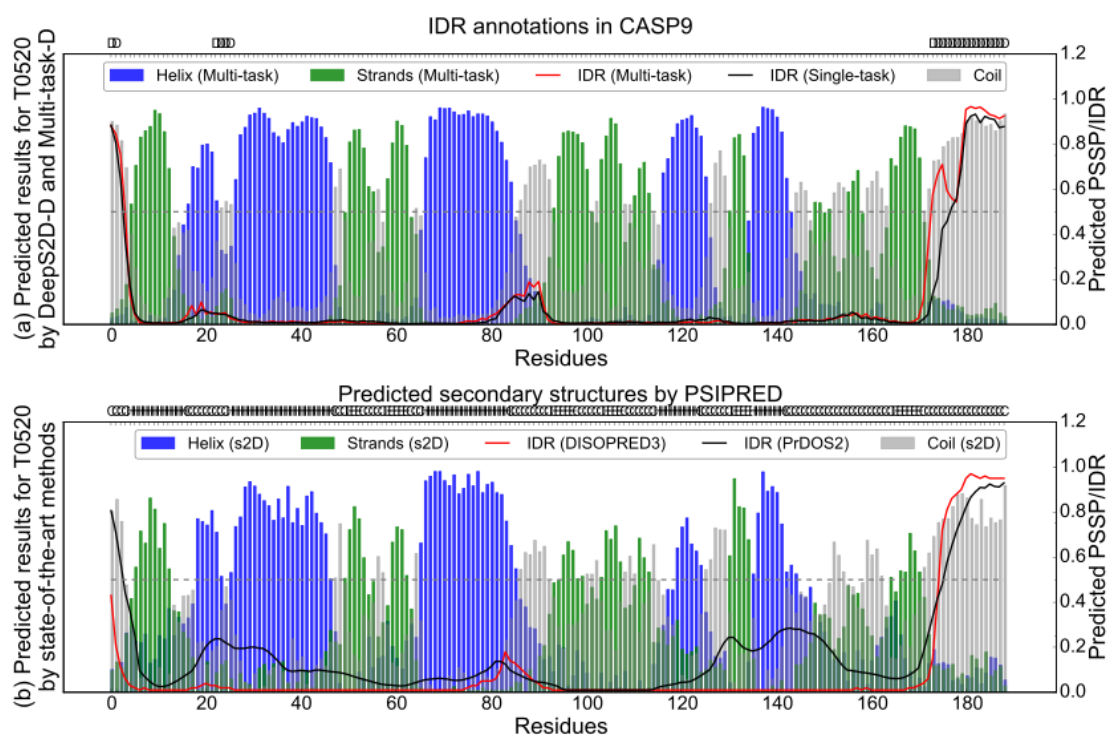
**Figure 6.3** Cross-stitch units indicating a linear correlation between IDP/IDR prediction (task A) and SSP prediction (task B).

The heat map in **Figure 6.3** is divided into four areas: cross\_stitch\_unit(AA), cross\_stitch\_unit(BA), cross\_stitch\_unit(AB) and cross\_stitch\_unit(BB), corresponding respectively to  $\alpha_{AA}$ ,  $\alpha_{BA}$ ,  $\alpha_{AB}$  and  $\alpha_{BB}$  in the cross-stitch units  $M$ , where task A represents IDP/IDR prediction and task B represents SSP prediction. Here,  $\alpha_{AA}$  and  $\alpha_{BB}$  connect the outputs from previous layers of the same task, while  $\alpha_{BA}$  and  $\alpha_{AB}$  connect the outputs from previous layers of the other task. According to the correlation heat map,  $\alpha_{AA}$  and  $\alpha_{BB}$  are darker than  $\alpha_{BA}$  and  $\alpha_{AB}$ , indicating that both tasks rely mainly on the outputs of its own previous layer. With the hidden layers going deeper, the dependency on the other task increases and the dependency on the same task decreases. It indicates that layers closer to the input layer tend to be more task-specific, while layers closer to the output layer are more likely to be affected by the other task.

#### 6.4.5 Case study on T520, p53 and SOSSB1

To better understand the mutual support between SSP and IDP/IDR prediction, we plotted the predicted results of DeepS2P-D, Multi-task-D, s2D [194], PSIPRED [104], PrDOS2 [211] and DISOPRED3 [101] for target T0520 from CASP9 in **Figure 6.4**. In this figure, both (a) and (b) show the results for the protein T520 from the CASP9 benchmark

dataset. **Figure 6.4** (a) demonstrates the IDP/IDRs (the black line) predicted by the single-task framework DeepS2P-D, the IDP/IDRs (the red line) and the SSPs (green, blue and grey bars for helix, strands and coils) predicted by Multi-task-D, the IDP/IDRs (the 'D' symbol in the top axis) annotated in the CASP9 benchmark dataset. In order to compare the predicted results of DeepS2P-D and Multi-task-D to that of existing methods, in **Figure 6.4** (b), we demonstrated the IDP/IDRs (the black line) predicted by PrDOS2, the IDP/IDRs (the red line) predicted by DISOPRED3, the SSPs (green, blue and grey bars for helix, strands and coils) predicted by the s2D method, and the secondary structures (the 'H(elix)', 'E(strands)', and 'C(oils)' labels in the top axis) predicted by PSIPRED.



**Figure 6.4** Prediction results for target T0520 in CASP9 by DeepS2P-D, Multi-task-D, PSIPRED, DISOPRED, PrDOS2 and s2D.

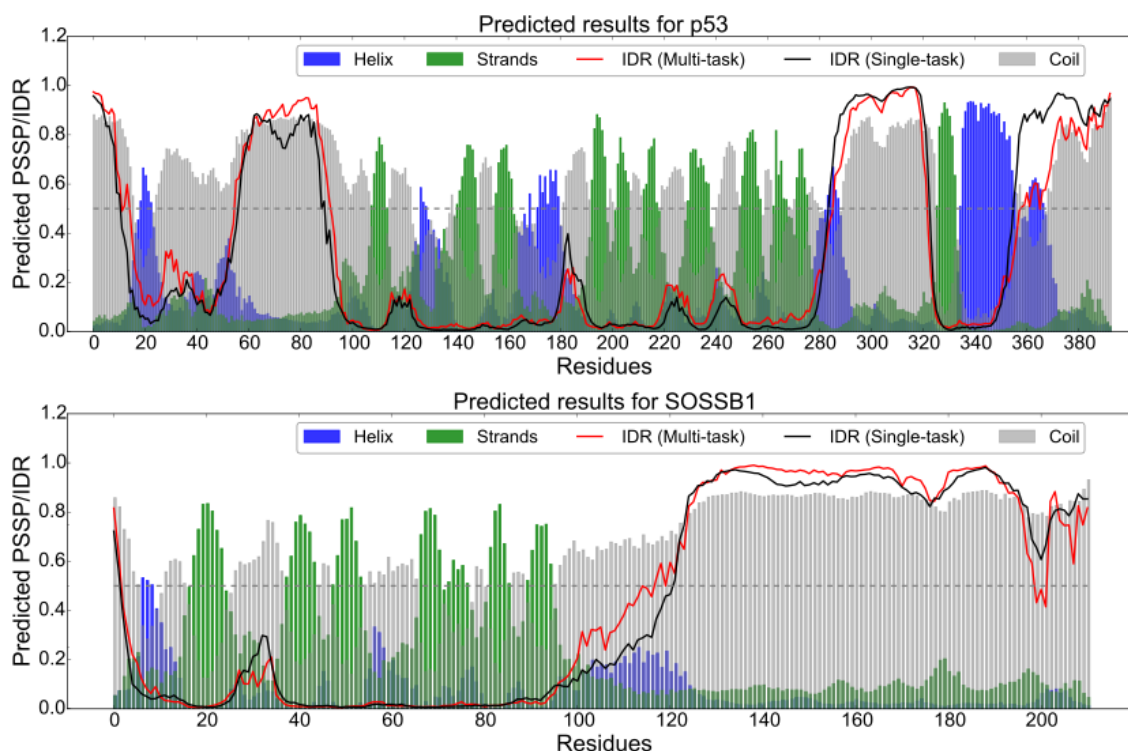
According to **Figure 6.4** (a), both DeepS2P-D and Multi-task-D predicted the first short IDR (residues 1-2) with one false positive prediction at residue 3 and failed to predict the second short IDR (residues 23-26). The third long IDR (residues 174-189)

was predicted by Multi-task-D with full accuracy, while DeepS2P-D predicted only 12 of the 16 disordered residues. This difference can be explained by the additional support obtained from the simultaneously predicted higher coil populations for residues 172-189 (indicated by 'grey' bars), which is only available in the Multi-task-D model. This, altogether, shows the benefits of using a multitask framework over a single-task framework.

In **Figure 6.4 (b)**, the SSP prediction results of the s2D method, Multi-task-D and the SS prediction results of PSIPRED generally agree with each other. In IDP/IDR prediction, DISOPRED3 missed the first short IDR and the first two residues in the third long IDR while PrDOS2 missed the first three residues in the third long IDR.

We further applied DeepS2P-D and Multi-task-D to protein p53 and SOSSB1 selected from the DisProt 7.0 database [89]. The prediction results are plotted in **Figure 6.5**. We discuss the respective results for these two proteins below.

**p53.** The p53 protein is composed of three functional regions, including the N-terminal region, the core DNA-binding region and the C-terminal region [329]. The N-terminal region is further divided into a transaction-activation domain (TAD, residues 1-63) and a proline-rich area (residues 64-93) [329]. The C-terminal region is further divided into a tetramerisation domain (residues 320-356) and a regulatory domain (residues 363-393) [330]. Multi-task-D predicted four disordered regions, where region 1 (residues 1-15) and region 2 (residues 56-92) are located in the N-terminal region, and region 3 (residues 287-322) and region 4 (residues 359-393) in the C-terminal region.



**Figure 6.5** Prediction results for p53 and SOSSB1 by DeepS2P-D and Multi-task-D.

According to [329], a) the whole N-terminal region p53(1-93) is disordered and b) residues 21-25 form a residual  $\alpha$ -helical segment, which is consistent with the known propensity of residues 18-25 to form an  $\alpha$ -helix when binding to MDM2 [331]. The first two predicted IDRs by Multi-task-D do not cover the whole N-terminal region, but the predicted SSPs in the gap region 18-23 reveals a larger helix population, which is consistent with b). According to the correlation between IDPs/IDRs and SSPs, a higher helix population may explain the predicted structure states in this region.

Another observation in the N-terminal region is that the second IDR (residues 56-92) predicted by Multi-task-D corresponds to the proline-rich domain (residue 64-93). According to [332], no significant chemical shifts were observed in this proline-rich domain, but resonances undergoing significant chemical shift changes were observed in the segment (residues 18-57), which corresponds to the structured region (residues 16-55) that was predicted by Multi-task-D. These corresponding regions suggest that

segments with significant chemical shift changes are less likely to be predicted as an IDR, which in turn explains the predicted structure states in residues 16-55.

The third predicted IDR (residue 287-322) was validated by the crystal structure of the core domain of p53 introduced in [333]. According to this crystal structure of p53, residues 278-289 form a  $\alpha$ -helix segment H2 for which the Multi-task-D model predicted an increase of helix population from 0.137 to 0.668 followed by a decrease to 0.471. With the decrease of helix population and the increase of the coil population, the H2 segment ends and, according to [333], residues up to Thr-312 are disordered. This disordered region, ranging from the end of H2 at residue 289 to residue Thr-312, overlaps with the third predicted IDR (residues 287-322).

Finally, the whole C-terminal region of p53 was annotated as a disordered region in DisProt 7.0. However, the Multi-task-D model predicted the regulatory domain (residues 363–393) to be disordered and the tetramerisation domain (residues 320-356) to be populated with strands and helices. This observation is consistent with the results in [329], showing that the tetramerisation domain of p53 adopts a well-defined conformation and is a folded domain. Our predictions validated these results.

**SOSSB1.** According to [334], residues 5-109 in SOSSB1 are composed of 6  $\beta$ -strands where  $\beta_1$ ,  $\beta_3$ ,  $\beta_4$ ,  $\beta_5$  and  $\beta_6$  form a typical OB fold and  $\beta_2$  is a small extra  $\beta$  strand that is antiparallel to  $\beta_3$ . This observation is validated by the 6 peaks of predicted strand populations in SOSSB1 (15-100) by the Multi-task-D model. The longest IDR (residues 110-211) is also predicted by DeepS2P-D and Multi-task-D. Other IDRs of SOSSB1 are relatively shorter, including residues 11-18, residues 28-37 and residues 71-81. For these shorter regions, despite that higher coil populations were predicted in residues 11-16 and residues 26-37, the IDR prediction scores did not surpass the cut-off threshold of 0.5. This indicates that if the predicted coil population is in a short region, this region is less likely to be predicted as an IDR. In another word, it indicates that the predicted IDRs are more likely to co-occur in long regions with predicted high coil populations.



#### 6.4.6 Conclusion

In this section, we simultaneously predicted IDPs/IDRs and SSPs by exploring the mutual correlation between these two tasks, using multitask deep learning neural networks. The cross-validation and independent test results demonstrate that the deep learning model DeepS2P-P outperforms the s2D method for predicting protein SSPs and that the representations learned for SSP and IDP/IDR prediction are mutually supportive.

However, despite the improved performance in both IDP/IDR and SSP prediction using multitask deep learning frameworks, the frameworks presented here can be extended in several ways. Firstly, additional protein features can be incorporated to improve the prediction performance. Other features that have proved useful for IDP/IDR prediction include physicochemical properties, structural features, and evolution-based features. Secondly, the Multi-task framework can be modified to automatically learn non-linear correlations among multiple tasks. The current cross-stitch framework only explores the linear relations between multiple tasks, which is a relatively simple assumption of the correlation among different protein sequence-based predictions. Third, other related tasks can be added to the multitask frameworks.

### 6.5 Chapter summary

In the chapter, we applied the DeepS2P framework to two protein structure properties, *i.e.* the protein intrinsic disorder (using the DeepS2P-D model) and protein secondary structure populations (using the DeepS2P-P model). We conducted comprehensive comparison experiments to determine the consolidated model structure of DeepS2P-D and DeepS2P-P. In IDP/IDR prediction, the positive effect of using deep learning architecture was demonstrated in two set of results. Firstly, during the process of hyperparameter tuning, average accuracy was improved from 0.915 to 0.923 when the hidden layers were improved from 1 to 3 (**Table 6.1**). Secondly, when comparing existing machine learning methods for IDP/IDR prediction on CASP9 and CASP10 targets, the

prediction performance of deep architectures in general were superior compared to those achieved by shallow architectures (**Table 6.2** and **Table 6.3**).

In SSP prediction, we compared the performance of linear models (the linear regression model), non-linear shallow models (the s2D method) and deep learning models (the DeepS2P-D model). We found that the DeepS2P-D model significantly improved the prediction performance compared to the previous two, which demonstrated the effectiveness of the DeepS2P framework in protein structure prediction tasks.

Based on the validation results of DeepS2P-P on disordered proteins and regions, we found that there exists close correlation between protein intrinsic disorder and secondary structure populations. It also validated existing experimental observations between these two protein structural properties (see **Section 4.3**). Therefore, we applied the proposed multitask frameworks to simultaneously predict IDP/IDR and SSP. By using SSP prediction as the supporting task for IDP/IDR prediction, we constructed the Multi-task-D model and the Cross-stitch-D model. Results in cross-validation test and independent tests demonstrated that Multi-task-D and Cross-stitch-D achieved superior performance compared to DeepS2P-D. Moreover, we provided case studies by applying the multitask prediction models to protein p53 and SOSSB1, based on which we observed supporting evidence of IDP/IDR prediction from the simultaneously predicted SSPs. Finally, using the Cross-stitch-D model, for the first time, we generated the quantitative correlation between the predictions of IDP/IDRs and SSPs.

# 7 Phosphorylation site prediction

## 7.1 Introduction

In **Section 4.1.4**, **Section 4.2** and **Section 4.4**, we introduced the application framework of context2vec for residue-level predictions, the CNN-based deep learning framework for predicting residue-level functional properties, and the deep transfer learning framework for predicting functionally important sites of enzymes that are organized in hierarchical structure. In this chapter, we introduce the application of these three deep learning techniques and frameworks to phosphorylation site prediction.

## 7.2 Protein kinase and its phosphorylation sites

Phosphorylation is the most common type of post-translation modification (PTM), which plays an essential role in the many cellular processes in eukaryotes [335, 336]. It is estimated that there are 13 000, 11 000, and 3000 phosphoproteins and 230 000, 156 000, and 40 000 phosphorylation sites exist in human, mouse, and yeast, respectively [337]. Phosphorylation refers to the post-translational modification where *the phosphoryl group* is covalently bound to amino acid residues such as serine (S), threonine (T) and tyrosine (Y), histidine (H) and lysine (amide bond), cysteine (thioester bond) or glutamic and aspartic acid (mixed anhydride bond) [34]. It involves the transfer of *the phosphoryl group* (e.g.  $\gamma$ -PO<sub>32</sub><sup>-</sup>) from the universal phosphoryl donor *adenosine triphosphate (ATP)* to the side chain of corresponding amino acids such as S, T, Y and H [338], resulting the *adenosine diphosphate (ADP)* which losses the phosphoryl group and the phosphoserine which gains the phosphoryl group. Residues of amino acid type S, T, Y or H, the sites on which phosphorylation happens, are referred to as *phosphorylation sites* or *phosphosites*.

Phosphorylation is chemically catalysed by protein kinases, a group of enzymes that facilitates the transfer of the phosphate group to substrates<sup>34</sup> [338]. Here, the *substrates* of phosphorylation refer to proteins that protein kinases catalyse. According to KinBase, more than 500 protein kinases have been identified in the human genome, which was organised into a classification tree of kinase subfamilies, families, and groups [300].

Different protein kinases specifically catalyse a group of substrates that matches its specificity criteria. The specificity of protein kinases relies on the combination of a number of independent and imperfect specificity mechanisms, including the structure of the catalytic site, local and distal interactions between the kinase and substrate, the formation of complexes with scaffolding and adaptor proteins that spatially regulate the kinase, systems-level competition between substrates, and error-correction mechanisms [335]. Therefore, phosphorylation is, to some extent, kinase-specific.

The experimental techniques for the identification of phosphosites include *Two-dimensional gel electrophoresis (2D gel)* [339] and *large-scale mass spectrometry* [55]. However, due to the expensive nature of these experimental methods, computational methods for predicting phosphorylation sites at a large scale are needed.

In this chapter, we introduce phosphorylation site prediction based on deep learning techniques. Firstly, we evaluate the proposed distributed representation of residue-level sequence contexts, *context2vec*, as a complementary input feature for biological representations in phosphorylation site prediction. Secondly, we demonstrate the limitations of applying the DeepS2P framework to phosphorylation sites of kinases with limited training data. Finally, we apply the DeepTransfer framework to predict phosphorylation sites of protein kinases that are organized in a four-level hierarchical classification system.

### 7.3 An overview of phosphorylation site prediction

---

<sup>34</sup> A substrate refers to the objects upon which the catalyse reaction happens.

The accurate prediction of kinase-mediated phosphorylation sites is important for functional annotations of target substrates and the elucidation of cellular signalling pathways underlying such phosphorylation events.

According to previous studies, the prediction of phosphorylation sites is addressed in two ways, *i.e. general phosphorylation site prediction and kinase-specific phosphorylation site prediction*. The former only considers the difference between different phosphorylation site types, for example, serine, threonine and tyrosine phosphorylation sites [216, 221], while the latter distinguishes different kinase-regulated phosphorylation sites in a kinase-specific manner [77, 214, 216, 218, 219, 340].

In general phosphorylation site prediction, the three most commonly phosphorylated sites were considered, including the serine (S), threonine (T) and tyrosine (Y) sites. Considering the different local sequence patterns demonstrated in these three kinases phosphorylation sites, three independent binary classifiers are built to predict whether a S, T or Y site is phosphorylated, respectively. Therefore, the general phosphorylation site prediction for target residue  $r_i$  can be formalised as follows,

$$M_s: f(v_{r_i}) \rightarrow \begin{cases} 1 \dots \text{if } r_i \text{ is phosphorylated.} \\ 0 \dots \text{if } r_i \text{ is not phosphorylated} \end{cases} \quad [\text{Eq. 7.1}]$$

where  $s \in S, T, Y$ .

In kinase-specific phosphorylation site prediction, a common assumption is that phosphorylation sites of the same protein kinase share similar sequential or structural patterns. Accordingly, independent binary classifiers are constructed for each kinase. Therefore, the kinase-specific phosphorylation site prediction is formalized as follows,

$$M_k: f(v_{r_i}) \rightarrow \begin{cases} 1 \dots \text{if } r_i \text{ is phosphorylated} \\ 0 \dots \text{if } r_i \text{ is not phosphorylated} \end{cases} \quad [\text{Eq. 7.2}]$$

Where  $k \in \text{PKA, PKC, CDK, CK2, SRC, ...}$ . Here, PKA, PKC, CDK, CK2 and SRC are some of the most widely researched protein kinases.

## 7.4 Phosphorylation site prediction based on context2vecs

In **Section 4.1.4**, we introduced the application of the distributed representation of protein sequences in three different scenarios. In this section, to validate the effectiveness of context2vec in residue-level predictions, we systematically evaluate and analyse the performance of context2vec in the task of phosphorylation site prediction, termed *PhosContext2vec*. More specifically, to take into consideration the dependence between phosphorylation sites and their sequential contextual patterns, we generated the distributed representation of the local sequence context for each potential phosphorylation site and then used it as a contextual feature vector to predict both general and kinase-specific phosphorylation sites. In particular, *PhosContext2vec* enables the representation of more specific contextual information than sequence-level features (such as ProtVec) [193] and provides contextual information complementary to other commonly used residue-level features, such as position-specific scoring matrixes and secondary structures [8, 99, 101].

Specifically, we propose two different implementation strategies to generate distributed contextual feature vectors for phosphorylation site prediction and compared their predictive performance. Firstly, we apply the word2vec-based summation implementation, *i.e.* ProtVec [193], to residue-level sequence contexts, referred to as context2vec<sup>add</sup> in this paper. Secondly, we apply the doc2vec model [169] to infer the distributed representation of residue-level sequence contexts directly from a pre-trained distributed memory network, referred to as context2vec<sup>inference</sup>. The major difference between context2vec<sup>add</sup> and context2vec<sup>inference</sup> is that the latter considers the order of amino acid sequences in the context of each residue, while the strategy of representation summation used by the former implementation does not consider this important aspect. Nevertheless, a disadvantage of context2vec<sup>inference</sup> is that the quality of the inferred representations may decrease slightly during the process of approximate inference.

In this section, we applied both context2vec<sup>add</sup> and context2vec<sup>inference</sup> to generate the prediction models of phosphorylation sites, benchmarked the performance of PhosContext2vec for both general and kinase-specific phosphorylation site predictions, and constructed the corresponding best-performing models for each type of phosphorylation sites and kinase families.

#### 7.4.1 Selected residue-level features for phosphorylation site prediction

We constructed models for general and kinase-specific phosphorylation site prediction using the Support Vector Machine (SVM) algorithm [145]. For each of the three phosphorylation site types S, T, and Y, and each of the five kinase families *AGC/PKC*, *AGC/PKA*, *Other/CK2*, *CGMC/CDK* and *TK/SRC*, an independent SVM model was constructed based on the proposed contextual feature vector in conjunction with six other residue-level feature groups. The contextual feature vector was generated from the aforementioned distributed representation of protein sequences and segments. The six residue-level features are the Shannon entropy, the relative entropy, protein disordered property, secondary structures, Taylor's overlapping properties and average cumulative hydrophobicity, which are briefly described below.

1. The Shannon entropy. It is a feature used for quantifying the conservation of potential phosphorylation sites [341]. It is calculated based on the weighted observed percentage (WOP), which can be generated by PSI-BLAST [99];
2. The relative entropy. It measures the conservation of amino acids compared to the background distributions, for example, BLOSUM62 [342]. It is also calculated based on the WOP;
3. Protein disorder information (DISO). Certain regions of the protein do not form stable structures and are called disordered regions. Several previous studies indicate that incorporating protein disorder information is useful for improving the prediction performance [298, 340, 343-346]. In this study, the protein disorder information was predicted using the DISOPRED3 program [101];

4. Protein secondary structure (PSS). It includes helix, strands and coil assignments in proteins. We used the tool package PSIPRED [104] to predict the secondary structure information for each protein sequence;
5. The Taylor's overlapping property (OP) .. It describes amino acid groups with respect to physiochemical properties. Each amino acid is encoded into 10 bits, each representing one of 10 physiochemical properties [347];
6. The average cumulative hydrophobicity (ACH). A varying sliding window of sizes 3, 5, 7, ..., 21 is used to extract the local sequence environment surrounding a potential phosphorylation site. The average hydrophobicity of all amino acids in the local window is calculated [203]. In this study, the Sweet and Eisenberg hydrophobicity index [348] was used to encode this feature group.

#### 7.4.2 Generation of context2vecs for phosphorylation site prediction

The generation of contextual feature vector was based on the distributed representation of protein sequences introduced in **Section 4.1**. More specifically, we generated the distributed representation of residue-level sequence contexts within specific contextual window sizes. This residue-level sequence context was referred to as  $PSP(m, n)$  in previous studies, namely the Phosphorylation Site Peptide, which was composed of  $m$  upstream residues and  $n$  downstream residues of the target sites [215]. In this study, we only considered the situation where an equal number of residues from the upstream and downstream of the target site was used, *i.e.*  $m = n$ . Thus, the contextual feature vector for a potential phosphorylation site  $r_i$  can be computed as follows,

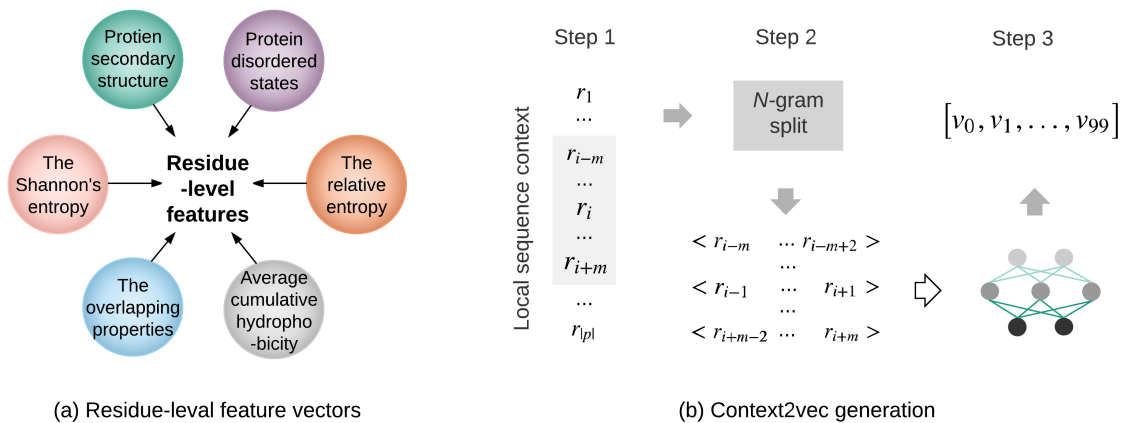
1. We extracted the contextual window of size  $w_i = 2m + 1$  for a residue  $r_i$ , which consists of  $m$  upstream residues and  $n = m$  downstream residues. The resulting contextual window was denoted as  $c_i = r_{i-m}, r_{i-m+1}, \dots, r_i, \dots, r_{i+m-1}, r_{i+m}$ ;



2. We performed  $n$ -gram split of the contextual window  $w_i$ , resulting in a list of biological words;
3. We generated the distributed representation of  $w_i$  in two different ways: (i) by summing up the distributed representation of all its biological words, named  $\text{context2vec}^{\text{add}}$ , and (ii) by feeding the list of biological words to the pre-trained distributed memory network  $\mathcal{N}$ , named  $\text{context2vec}^{\text{inference}}$ .

The final feature vector of the potential phosphorylation site  $r_i$  is constructed by stacking the above seven groups of feature vectors together, resulting in a 126-dimensional feature vector.

**Figure 7.1** (a) illustrates the six groups of residue-level input features generated from homology profiles, physicochemical properties and structural information, and **Figure 7.1** (b) provides a flowchart of how to generate the purely data-driven representation,  $\text{context2vec}$ , for local sequence contexts.



**Figure 7.1** Extraction of residue-level feature groups and the generation of the distributed contextual feature vector for phosphorylation site prediction.

### 7.4.3 Experimental design

**Datasets.** To assess the performance of PhosContext2vec in comparison with other existing methods, we conducted cross-validation and independent tests in both general and kinase-specific phosphorylation site predictions.

For general phosphorylation site prediction, we used the same training and independent test datasets of annotated Serine (S), Threonine (T) and Tyrosine (Y) phosphorylation sites that were introduced in PhosphoSVM [221]. The training dataset, termed PELM, was constructed from the database Phospho.ELM (Version 9.0) [94] which includes experimentally verified phosphorylation sites in animals such as *Homo sapiens*, *Mus musculus*, *Drosophila melanogaster* and *Caenorhabditis elegans*. It has 6632, 3226 and 1,392 proteins containing 20,960, 5,684 and 2,163 phosphorylation sites for S, T, and Y sites respectively. The independent test set, named PPA, was extracted from the database PhosphoAt (Version 3.0) [349] which only contains plant phosphorylation sites from *A. thaliana*. The PPA dataset contains 3,037, 1,359 and 617 substrate proteins with 5,449, 1,686 and 676 annotated phosphorylation sites, for S, T, and Y sites respectively.

For kinase-specific phosphorylation site prediction, we combined phosphorylation site data obtained from Phospho.ELM [94] and Swiss-Prot/UniProt [39] and constructed training and independent test datasets in following steps. Firstly, we downloaded 555,594 reviewed proteins from Swiss-Prot/UniProt and extracted all the proteins that had at least one phosphorylation site annotation, resulting in 14,458 proteins in total. We then collected triplet-record annotations (protein, phosphorylation site position, and kinase) from Swiss-Prot/UniProt for the 14,458 proteins and removed 2,155 triplet-record annotations that were labelled as 'by similarity' in UniProt. The resulting 56,772 triple-record annotations contained 43,785 phosphorylated S sites, 10,397 phosphorylated T sites, and 4,711 phosphorylated Y sites, among which 7,021, 2,515 and 2,066 were respectively annotated with kinase types. Similarly, we extracted the triple-record

annotations from the Phospho.ELM (version 9.0) database, resulting in 43,027 phosphorylated S sites, 9,556 phosphorylated T sites, and 4,723 phosphorylated Y sites, among which 2,961, 943, and 1,031 sites were respectively annotated with corresponding kinase types.

In order to combine triple-record annotations from Swiss-Prot/UniProt and Phospho.ELM, we employed the following procedures.

- (a) We cross-checked and renamed proteins extracted from the Swiss-Prot/UniProt database whose sequences have been updated compared to those in the Phospho.ELM database;
- (b) We manually corrected the kinase names according to the hierarchical structure of the kinase groups, families, subfamilies, and types (Please refer to Table S1 in the GPS 2.0 paper [214]);
- (c) We removed redundant entries that were included in both two databases; and
- (d) We excluded kinases that had less than 20 triple-record annotations. In total, we obtained consolidated phosphorylation sites for 138 kinase groups, families, subfamilies and protein kinases.

In this study, we only performed cross-validation and independent tests on five kinase families that had more than 500 triple-record annotations. These included AGC/PKC, AGC/PKA, Other/CK2, CGMC/CDK, and TK/SRC with 962, 897, 668, 628, and 631 triple-record annotations, respectively. To obtain the training and testing datasets, we divided the triple-record annotations into five subsets, among which four were used as the positive samples in training data and the remaining one was used as the positive samples in testing data. At the same time, S, T, and Y sites that were not annotated as phosphosites were treated as negative samples. A statistical summary of the finally curated datasets is shown in the **Appendix A**.

**Table 7.1** Statistical summary of the curated datasets for the five kinases with the most number of annotations.

		Training	Testing
AGC/PKA	#pos	716	181
	#neg	58,967	15,288
AGC/PKC	#pos	766	196
	#neg	41,473	10,672
CMGC/CDK	#pos	486	142
	#neg	34,192	7,443
Other/CK2	#pos	532	136
	#neg	20,564	6,167
TK/Src	#pos	496	135
	#neg	29,591	8,246

The curated phosphorylation site datasets are highly imbalanced, whereby the negative examples are hundreds of times more than the positive examples. In previous studies, the issue of imbalanced datasets was addressed by down-sampling the negative examples [215, 216, 218, 221] or augmenting the positive examples [35]. For down-sampling, the negative examples can be randomly sampled from all [S, T and Y] sites that were not annotated as phosphorylation sites [215, 218, 221]. They can also be sampled from non-phosphorylated S, T or Y sites depending on the type of site or kinase the model was trained for [216]. In this paper, we included all [S, T, and Y] sites, which were not annotated as phosphosites, as negative examples for the down-sampling performed in the training set, reducing the number of negative examples to be equal to the number of positive examples. In the independent test set, we included S, T or Y sites depending on the type of sites or the kinase the model was trained for, without down-sampling. For example, for the model that was trained to predict phosphorylated Y sites, we only included all Y sites, which were not annotated as phosphosites, as negative examples. While for the model that was trained to predict sites that are specifically phosphorylated by the AGC/PKA kinase, we included non-phosphorylated S and T sites as negative examples.

**Cross-validation and independent tests.** We performed 10-fold cross-validation tests using each of the aforementioned benchmark datasets where, in each iteration, nine folds were used for training the model while the remaining fold was used to validate the prediction performance of the trained model. This process was repeated 10 times so that each fold was used for both model training and validation. Based on 10-fold cross-validation results, we selected the model that achieved the best performance on the validation set and further tested its performance on the independent test datasets. Different from the cross-validation for which down-sampling was used to ensure selection of equal numbers of positive and negative examples, independent tests were performed against imbalanced datasets which contained more negative examples than positive examples. Therefore, although the performance in independent tests appeared to be worse than that in cross-validation tests, it resembled a real-world scenario more closely.

**Performance evaluation.** We evaluated the prediction performance of constructed models using four measurements, including sensitivity, specificity, the Matthews coefficients of correlation (MCC) and the area under the ROC curve (AUC) (Please refer to **Appendix A** for detailed equations). Except for AUC, all the other three measures rely on the selection of the prediction cut-off thresholds for models to generate the final classification outcome. Because different predictors produced different ranges of predicted probability scores, we used the False Positive Rate (FPR) to determine positive predictions. According to GPS 3.0 [77], the low, medium and high cut-off FPRs for S and T sites were respectively set as 2%, 6% and 10%, while the low, medium and high cut-off FPRs for Y sites were respectively set as 4%, 9% and 15%.

#### **7.4.4 The effect of incorporating the distributed contextual feature vectors**

To characterize the effect of distributed representation of sequence contexts on both general and kinase-specific phosphorylation site prediction, we performed 10-fold cross-validation tests and evaluated the performance of models trained using residue-level

features only (Residue-level), the context2vec generated with the inference strategy ( $\text{Context2vec}^{\text{inference}}$ ), the context2vec generated with the addition strategy ( $\text{Context2vec}^{\text{add}}$ ), the combination of residue-level features and  $\text{context2vec}^{\text{inference}}$  ( $\text{Residue-level}+\text{Context2vec}^{\text{inference}}$ ), the combination of residue-level features and  $\text{context2vec}^{\text{add}}$  ( $\text{Residue-level}+\text{Context2vec}^{\text{add}}$ ), and the combination of residue-level features and ProtVec [193] ( $\text{Residue-level}+\text{prot2vec}^{\text{add}}$  (ProtVec)). For the combined feature vectors, they were constructed by concatenating two feature vector representations for each target residue.

**Table 7.2** shows the performance results of different models trained using different groups of input features for general and kinase-specific phosphorylation site prediction, respectively. For  $\text{Context2vec}^{\text{inference}}$ ,  $\text{Context2vec}^{\text{add}}$ ,  $\text{Residue-level}+\text{Context2vec}^{\text{inference}}$ , and  $\text{Residue-level}+\text{Context2vec}^{\text{add}}$  feature vectors, four different contextual window sizes (ws) 7, 11, 15, and 19 were used to extract the local sequence contexts based on which the context2vec features were generated. It should be noted that the Residue-level features were representative of the special case of contextual window size 0, while the  $\text{Residue-level}+\text{prot2vec}^{\text{add}}$  (ProtVec) represents the special case of infinite contextual window size (including the whole sequence).

**Table 7.2** Performance comparison between models trained with and without distributed contextual feature vectors for general and kinase-specific phosphorylation site prediction.

(a) Results for general phosphorylation site prediction

	ws	S	T	Y
Residue-level	0	0.842+/-0.008	0.896+/-0.007	0.933+/-0.007
Context2vec <sup>inference</sup>	7	0.512+/-0.005	0.557+/-0.016	0.854+/-0.036
	11	<u>0.845+/-0.005</u>	<u>0.861+/-0.011</u>	<u>0.871+/-0.016</u>
	15	0.841+/-0.006	0.848+/-0.010	0.849+/-0.017
	19	0.835+/-0.006	0.836+/-0.014	0.833+/-0.017
Context2vec <sup>add</sup>	7	<u>0.852+/-0.008</u>	<u>0.884+/-0.009</u>	<u>0.909+/-0.011</u>
	11	0.850+/-0.011	0.867+/-0.009	0.875+/-0.017
	15	0.843+/-0.013	0.853+/-0.011	0.856+/-0.018
	19	0.841+/-0.006	0.840+/-0.014	0.840+/-0.018
Residue-level + Context2vec <sup>inference</sup>	7	0.887+/-0.008	0.921+/-0.007	0.938+/-0.008
	11	0.889+/-0.008	0.926+/-0.008	0.938+/-0.008
	15	<b>0.889+/-0.008</b>	<b>0.927+/-0.009</b>	0.939+/-0.008
	19	0.887+/-0.008	0.926+/-0.008	<b>0.939+/-0.008</b>
Residue-level + Context2vec <sup>add</sup>	7	0.887+/-0.008	0.920+/-0.008	0.937+/-0.008
	11	0.892+/-0.008	0.927+/-0.007	0.938+/-0.008
	15	<b>0.892+/-0.008</b>	<b>0.929+/-0.006</b>	0.939+/-0.008
	19	0.891+/-0.008	0.929+/-0.005	<b>0.939+/-0.008</b>
Residue-level + prot2vec <sup>add</sup> (ProtVec)	inf	0.842+/-0.010	0.901+/-0.006	0.938+/-0.007

\* For each of the three sites, the best performance of models with combined input features (including Residue-level+ Context2vec<sup>add</sup> and Residue-level+ Context2vec<sup>inference</sup>) was respectively highlighted as bold, while the best performance of models with Context2vec-only input feature (including Context2vec<sup>add</sup> and Context2vec<sup>inference</sup>) was highlighted with underlines.

(b) Results for kinase-specific phosphorylation site prediction

	ws	AGC/PKA	AGC/PKC	CMGC/CDK	Other/CK2	TK/Src
Residue-level	0	0.915+/-0.013	0.898+/-0.022	0.799+/-0.040	0.839+/-0.040	0.956+/-0.011
Context2vec <sup>inference</sup>	7	0.609+/-0.043	0.653+/-0.056	0.570+/-0.042	0.736+/-0.056	0.828+/-0.051
	11	<u>0.918+/-0.013</u>	0.862+/-0.031	<u>0.890+/-0.029</u>	0.885+/-0.048	<u>0.913+/-0.014</u>
	15	0.895+/-0.020	0.862+/-0.028	0.885+/-0.030	<u>0.893+/-0.043</u>	0.895+/-0.016
	19	0.880+/-0.018	<u>0.865+/-0.026</u>	0.872+/-0.031	0.889+/-0.044	0.875+/-0.022
Context2vec <sup>add</sup>	7	0.885+/-0.025	0.809+/-0.022	<u>0.903+/-0.030</u>	0.814+/-0.055	<u>0.931+/-0.010</u>
	11	<u>0.925+/-0.013</u>	0.872+/-0.026	0.883+/-0.026	0.882+/-0.037	0.912+/-0.014
	15	0.906+/-0.015	0.882+/-0.023	0.872+/-0.034	0.898+/-0.032	0.893+/-0.017
	19	0.892+/-0.016	<u>0.885+/-0.021</u>	0.869+/-0.034	<u>0.901+/-0.032</u>	0.877+/-0.017
Residue-level + Context2vec <sup>inference</sup>	7	0.928+/-0.011	0.897+/-0.028	<b>0.925+/-0.018</b>	0.866+/-0.041	0.960+/-0.009
	11	<b>0.938+/-0.009</b>	0.909+/-0.027	0.908+/-0.018	0.907+/-0.037	0.964+/-0.008
	15	0.937+/-0.010	0.907+/-0.028	0.902+/-0.024	<b>0.919+/-0.034</b>	<b>0.966+/-0.007</b>
	19	0.937+/-0.011	<b>0.909+/-0.024</b>	0.899+/-0.025	0.917+/-0.033	0.965+/-0.008
Residue-level + Context2vec <sup>add</sup>	7	0.927+/-0.012	0.895+/-0.028	<b>0.907+/-0.020</b>	0.864+/-0.040	0.957+/-0.009
	11	0.939+/-0.010	0.911+/-0.023	0.896+/-0.020	0.911+/-0.034	0.962+/-0.009
	15	0.939+/-0.010	0.913+/-0.023	0.890+/-0.022	0.927+/-0.026	0.964+/-0.008
	19	<b>0.940+/-0.010</b>	<b>0.915+/-0.020</b>	0.894+/-0.024	<b>0.929+/-0.024</b>	<b>0.964+/-0.008</b>
Residue-level + prot2vec <sup>add</sup> (ProtVec)	inf	0.908+/-0.013	0.874+/-0.022	0.738+/-0.042	0.827+/-0.036	0.954+/-0.010

\* For each of the five kinases, the best performance of models with combined input features (including Residue-level+ Context2vec<sup>add</sup> and Residue-level+ Context2vec<sup>inference</sup>) was respectively highlighted as bold, while the best performance of models with Context2vec-only input feature (including Context2vec<sup>add</sup> and Context2vec<sup>inference</sup>) was highlighted with underlines.



In **Table 7.2**, the best predictive performance for each phosphorylation site type or kinase family achieved by Context2vec-only feature vectors (including Context2vec<sup>inference</sup> and Context2vec<sup>add</sup>) and Residue-level+Context2vec feature vectors (including Residue-level+Context2vec<sup>inference</sup> and Residue-level+Context2vec<sup>add</sup>) with respect to different contextual window sizes was highlighted by underline and bold, respectively.

As clearly shown in **Table 7.2** (a) and (b), the models trained using Residue-level+Context2vec features outperformed the models trained with Residue-level features across all three types of phosphorylation sites and five kinase families. For example, the models employing the combination of residue-level features and the context2vec<sup>inference</sup> (ws=7) improved the average AUC scores by 0.045, 0.025 and 0.005 for phosphorylation site prediction on S, T, and Y sites, and by 0.013, -0.002, 0.126, 0.027, and 0.004 for kinase AGC/PKA, AGC/PKC, CGMC/CDK Other/CK2, and TK/SRC, respectively, compared with the models trained with residue-level features only. The only exception was the AGC/PKC kinase, for which the model trained with context2vec<sup>inference</sup> of window size 7 performed slightly worse than the baseline model. However, the average AUC achieved with Residue-level+Context2vec<sup>inference</sup> feature vectors was increased from 0.897 to 0.909 when increasing the contextual window size from 7 to 19, resulting in better performance than that was achieved by models trained with residue-level features only. These improved results demonstrate that, for both general and kinase-specific phosphorylation site prediction, the prediction performance achieved with the Residue-level+Context2vec features could be further improved when the contextual window sizes of Context2vec features were appropriately selected. This was more pronounced for phosphorylation site types S and T, and the CMGC/CDK kinase, with the average AUC scores significantly increased by 0.047, 0.031 and 0.126 (significant level: 95%), respectively.

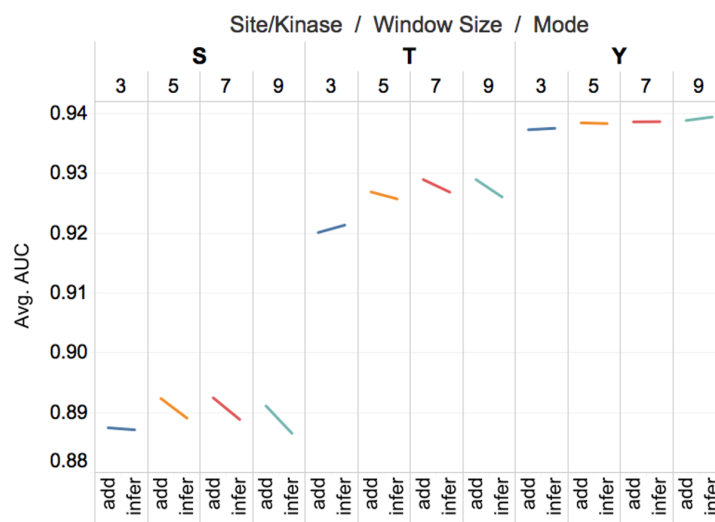
Comparing the predictive performance achieved by the models trained with Context2vec-only and Residue-level+Context2vec features, it can be observed that the models trained with the latter outperformed those trained with the former (**Table 7.2**). Taking kinase-specific prediction as an example, the best performance in terms of AUC scores was improved from 0.918, 0.865, 0.890, 0.893, and 0.913 to 0.938, 0.909, 0.925, 0.919, and 0.966 for AGC/PKA, AGC/PKC, CMGC/CDK, Other/CK2, and TK/Src, respectively, when Context2vec<sup>inference</sup> was used in combination with Residue-level features. Similar improvements can be observed by comparing the results achieved by Context2vec<sup>add</sup> and Residue-level+Context2vec<sup>add</sup>. For example, the best AUC scores for the five kinase families were improved from 0.925, 0.885, 0.903, 0.901, and 0.931 to 0.940, 0.915, 0.907, 0.929, and 0.964, respectively, when Context2vec<sup>add</sup> was used in combination with Residue-level features. The results suggest that the Context2vec, as a useful contextual feature vector, can achieve better predictive performance when used in combination with residue-level features.

In terms of the effect of contextual window size, the performance (evaluated in terms of AUC scores) depended on the phosphorylation site type and kinase family. In general, with the increase of the contextual window size, the AUC score tended to increase until reaching the peak and then started to decrease. For example, the average AUC score of the models trained using Residue-level+Context2vec features was improved from 0.887 to 0.892 and decreased to 0.891 when the contextual window size was increased from 7 to 15 and decreased from 15 to 19, respectively, for S phosphorylation site prediction. For CGMC/CDK, the best AUC score of 0.926 was achieved using the contextual window size of 7, and then decreased to 0.899 when the window size was increased to 19. As aforementioned, Residue-level features can be seen as the special case where the contextual window size is set as 0, whereas the sequence-level representation prot2vec<sup>add</sup> (*i.e.* ProtVec) is equivalent to the special case where the contextual window size is set as infinite.

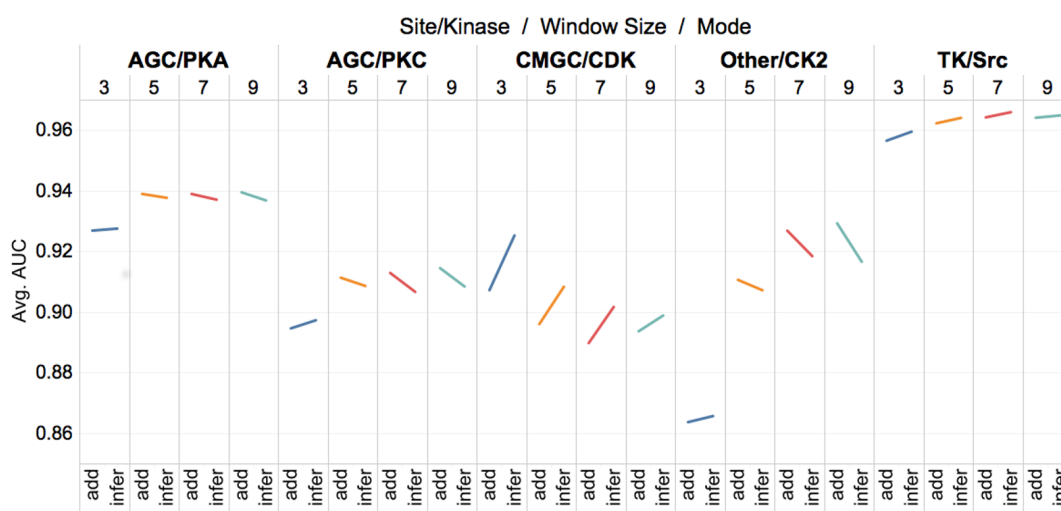
According to the results in **Table 7.2**, models that were trained with these two types of feature vectors achieved worse performance compared to that of the models trained with Context2vec features. These results highlight the need and importance of developing specialized distributed representation of local sequence contexts that are more informative to address the residue-level prediction tasks, such as phosphorylation site prediction in this study.

#### **7.4.5 The comparison between two implementations of prot2vecs**

We further compared the performance of the two models trained using two types of contextual feature vectors generated by context2vec<sup>add</sup> and context2vec<sup>inference</sup> based on the same group of results. **Figure 7.2** (a) and (b) plot the average AUC scores achieved by context2vec<sup>add</sup> and context2vec<sup>inference</sup> models for both general and kinase-specific phosphorylation site prediction. Models trained using different contextual window sizes 7, 11, 15, and 19 (denoted as 3, 5, 7, and 9, respectively) are indicated by different colours. For each short line, the left side indicates the average AUC score achieved by the context2vecadd model, while the right side indicates the average AUC score achieved by context2vecinference model. The tilt angle of each short line thus indicates the performance difference between the two contextual representation-based models.



(a) General phosphorylation site prediction



(b) Kinase-specific phosphorylation site prediction

**Figure 7.2** Performance comparison between models trained based on context2vec<sup>add</sup> and context2vec<sup>inference</sup> feature vectors for (a) **general** and (b) **kinase-specific** phosphorylation site prediction, evaluated in terms of AUC score.

According to **Figure 7.2**, two important observations can be made. The first observation is that, regardless of the selection of the contextual window size, one of the two implementations performed better than the other depending on the type of phosphorylation sites and kinase families. Specifically, in the case of S/T phosphorylation sites and most the of five kinases that phosphorylate S/T phosphorylation sites, models trained with  $\text{context2vec}^{\text{add}}$  performed better than those trained with  $\text{context2vec}^{\text{inference}}$ , while in the case of Y phosphorylation sites, the TK/SRC kinase that phosphorylates Y sites, and the CGMC/CDK kinase that phosphorylates S/T sites, models trained with  $\text{context2vec}^{\text{inference}}$  performed better than those trained with  $\text{context2vec}^{\text{add}}$ .

The second observation is that the performance of  $\text{context2vec}^{\text{add}}$  improved faster than that of the  $\text{context2vec}^{\text{inference}}$  with the increase of the contextual window size. This is manifested by the relative change of the tilt angle with respect to different contextual window size in **Figure 7.2**. For example, for Other/CK2 (represented by the skyblue lines in **Figure 7.2** (b)), the tilt angle had a transition from slightly favoring  $\text{context2vec}^{\text{inference}}$  to gradually rising up on the side of  $\text{context2vec}^{\text{add}}$  with the increase of the contextual window size. For CGMC/CDK, the performance of both  $\text{context2vec}^{\text{add}}$  and  $\text{context2vec}^{\text{inference}}$  decreased with the increase of the contextual window size, but the performance of  $\text{context2vec}^{\text{inference}}$  decreased faster than that of  $\text{context2vec}^{\text{add}}$ , resulting in shrinking of the tilt angle with the increase of the contextual window size. The performance difference between the two implementations of context2vecs validated their respective advantages and the disadvantages. On one hand, the  $\text{context2vec}^{\text{inference}}$  took into consideration the order of amino acids in protein sequences while  $\text{context2vec}^{\text{add}}$  ignores this important information. On the other hand, the quality of  $\text{context2vec}^{\text{inference}}$  may decrease in the approximation-based inference step, while the  $\text{context2vec}^{\text{add}}$  was generated by summing up the pre-trained representations of biological words that are directed fetched from the training step.

According to the AUC scores and their standard deviations shown in **Table 7.2** and **Figure 7.2**, we finally selected context2vec<sup>add</sup> with the window size of 15, context2vec<sup>add</sup> with the window size of 15, and context2vec<sup>inference</sup> with the window size of 19 for predicting S, T, and Y phosphorylation sites, respectively, and context2vec<sup>add</sup> with the window size of 19, context2vec<sup>add</sup> with the window size of 19, context2vec<sup>inference</sup> with the window size of 7, context2vec<sup>add</sup> with the window size of 19, and context2vec<sup>inference</sup> with the window size of 15 for predicting phosphorylation sites of AGC/PKA, AGC/PKC, CGMC/CDK, Other/CK2 and TK/SRC kinases, respectively. The results suggest that this new distributed representation of contextual features improved the prediction of both general and kinase-specific phosphorylation sites. According to the 10-fold cross-validation results, the overall predictive performance was improved by 0.051, 0.033, and 0.006 for the prediction of S, T, and Y phosphorylation sites, and improved by 0.025, 0.017, 0.127, 0.090, and 0.001 for the prediction of AGC/PKA, AGC/PKC, CMGC/CDK, Other/CK2, and TK/SRC phosphorylation sites, respectively, compared with those of the models that were trained with residue-level feature only.

#### 7.4.6 Determination of hyper-parameters

We trained independent SVM models for each of the three types of phosphorylation sites and five kinase families, for which two hyper-parameters would influence the prediction performance and thus needed to be determined. These two hyper-parameters were the kernel type, which determined the kernel that was used for training the model, and the penalty parameter C of the error term, which affected the trade-off between the complexity and proportion of non-separable examples [350].

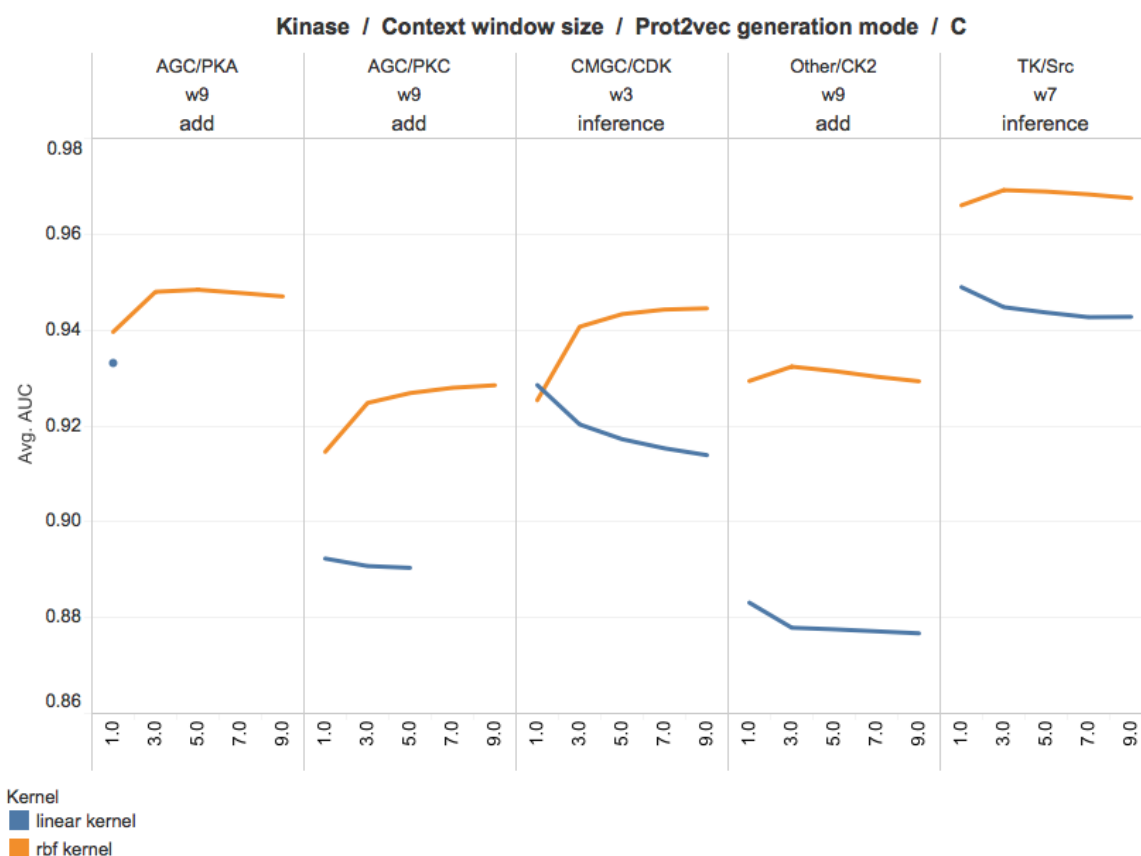
We tested the performance of the models trained with two most commonly used kernel types including the linear kernel and the RBF kernel [351], as well as five different values (including 1.0, 3.0, 5.0, 7.0 and 9.0) of the penalty parameter C [351]. Note that a larger penalty C indicates a stronger penalty on non-separable examples, which will

result in more complex models that fit the data more strictly. However, it may also cause overfitting at the same time.

Based on the optimal contextual window sizes and context2vec representation implementation selected from above experiments, we performed 10-fold cross-validation tests to compare the prediction performance between models trained with the different kernels types and five values of penalty parameter  $C$ . The corresponding results are shown in **Figure 7.3**, in which “add” and “inference” indicate the  $\text{context2vec}^{\text{add}}$  and  $\text{context2vec}^{\text{inference}}$  representations, respectively. “w” indicates the contextual window size. Note that the results of models that did not converge during training were not included in this figure.



(a) Hyper-parameter tuning in general phosphorylation site prediction



(b) Hyper-parameter tuning in kinase-specific phosphorylation site prediction

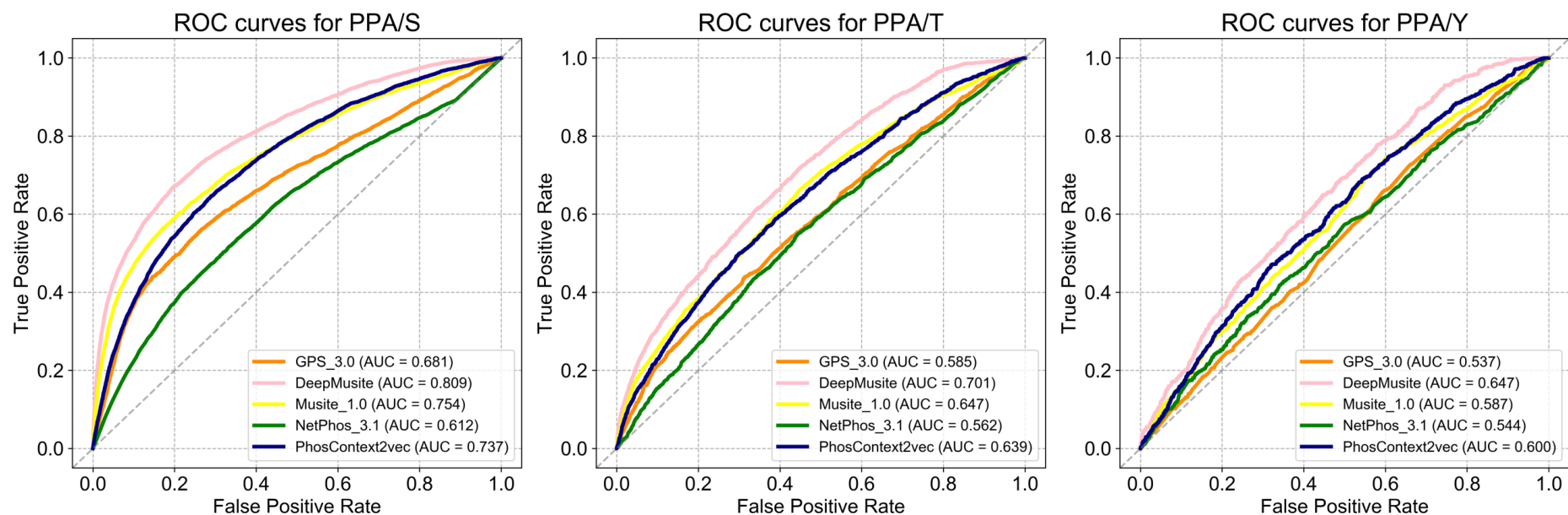
**Figure 7.3** Hyper-parameter tuning for SVM models trained with distributed contextual feature vectors for (a) **general** and (b) **kinase-specific** phosphorylation site prediction.



For general phosphorylation site prediction, the RBF kernel performed better for S phosphorylation sites, while the linear kernel performed better for T/Y phosphorylation sites. At the same time, the best performance for S, T and Y phosphorylation site prediction was achieved when the value of the penalty parameter C was set to 3.0, 7.0 and 7.0, respectively. For kinase-specific phosphorylation site prediction, the RBF kernel performed better than the linear kernel across all five kinase families. Accordingly, the best performance was achieved when the value of C was set to 5.0, 9.0, 9.0, 3.0 and 3.0 for the AGC/PKA, AGC/PKC, CMGC/CDK, Other/CK2 and TK/SRC kinase families, respectively.

#### **7.4.7 Independent test for general phosphorylation site prediction**

In order to validate the performance of PhosContext2vec for general phosphorylation site prediction, we further performed comprehensive independent tests. Using optimised contextual window size, context2vec representation implementations and hyper-parameters selected on the cross-validation, we trained the models for S, T, and Y sites using the PELM training datasets and then evaluated the performance using the curated PPA datasets. We submitted the same datasets to other existing predictors including GPS 3.0, Musite 1.0 and NetPhos 3.1 and collected the prediction results. **Figure 7.4** shows the ROC curves and corresponding AUC scores of all the five compared methods for general phosphorylation site prediction, for PPA/S, PPA/T and PPA/Y, respectively.



**Figure 7.4** ROC curves of PhosContext2Vec and four existing methods on the independent test for general phosphorylation site prediction.

According to the results shown in **Figure 7.4**, MusiteDeep achieved the best prediction performance in terms of AUC scores, and Musite 1.0 and PhosContext2vec outperformed the other two methods, GPS 3.0 and NetPhos 3.1. MusiteDeep achieved an overall best performance for S, T and Y phosphorylation site prediction with an AUC score of 0.809, 0.701 and 0.647, respectively, while Musite 1.0 achieved the second-best AUC scores of 0.754 and 0.647 for S and T phosphorylation site prediction and PhosContext2vec achieved the second-best AUC score of 0.600 for Y phosphorylation site prediction. PhosContext2vec achieved the third-best performance for S and T phosphorylation site prediction with an average AUC score of 0.737 and 0.639, respectively.

Among the five compared methods, MusiteDeep is the only method that employed the deep learning architecture. It takes raw amino acid sequences as the direct input, extracts high abstract representations from the 16 upstream and downstream residues using convolutional layers and the attention mechanism, and performs phosphorylation prediction based on the extracted representation [78]. In comparison, PhosContext2vec employed a more traditional way of decoupling the process of feature extraction and predictive model training and the algorithm of SVM which has less expressive power [3], which may explain the inferior performance of PhosContext2vec compared to that of the MusiteDeep. Nevertheless, the representation extracted in MusiteDeep purely rely on the neighbouring residues of the target residue, while the contextual feature vectors in PhosContext2vec also involve information that is distributed over large protein sequence databases, which represents its advantage as a contextual representation.

Compared with GPS 3.0 and NetPhos 3.1, both Musite 1.0 and PhosContext2vec considered protein disorder information as the input feature for training the models. Besides, Musite 1.0 also considered amino acid frequencies and used an ensemble learning strategy to make the final prediction, which might be superior to individual SVM model or neural network models. GPS 3.0 was designed to predict phosphorylation sites

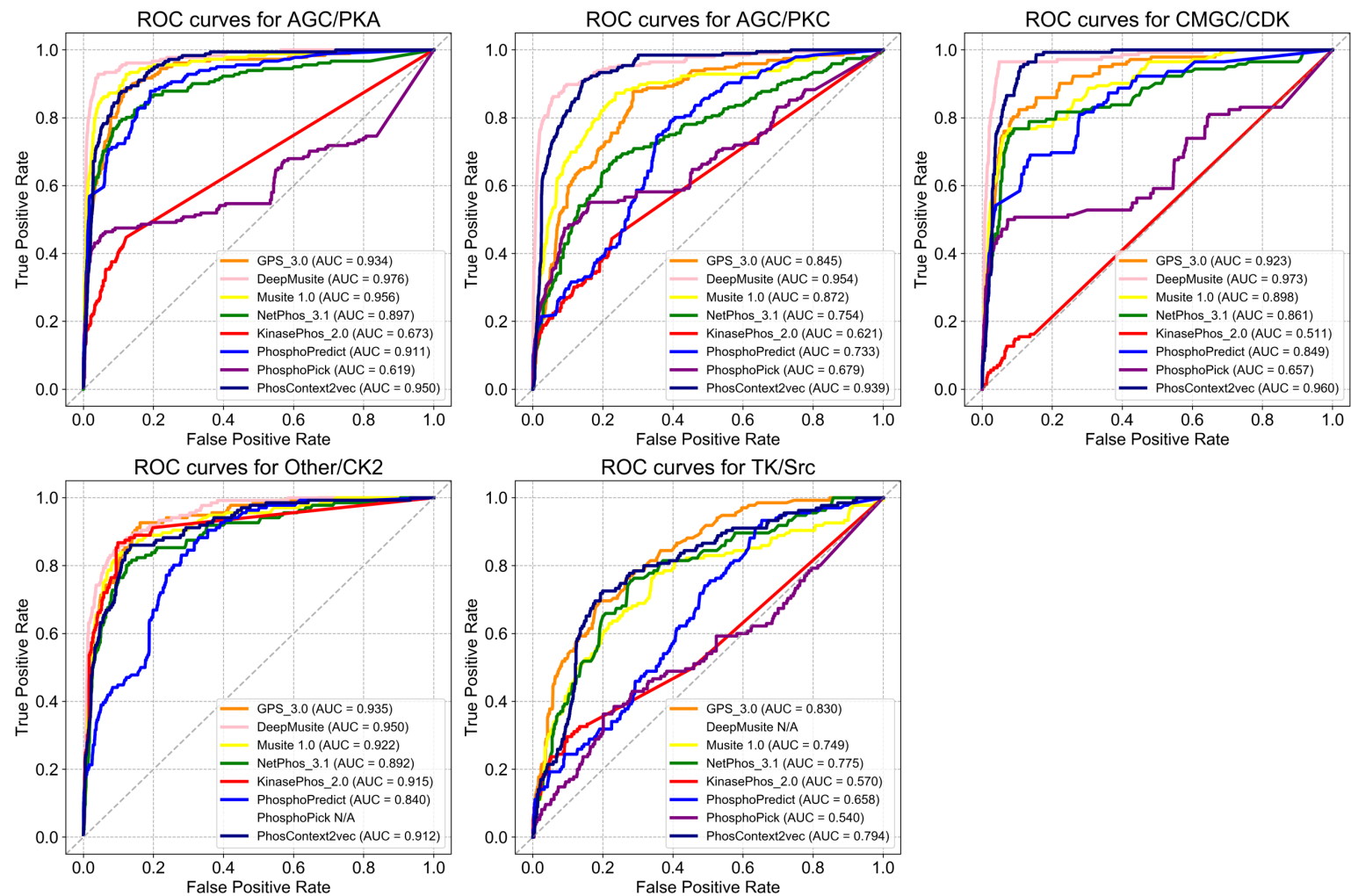
in a way that kinases were clustered in a hierarchical structure, where the training examples of related kinases can be reused for training better models. In the case of general phosphorylation site prediction, where models were respectively trained for S, T and Y sites (not in a hierarchical manner), the hierarchical clustering of kinases did not appear to help improve the performance. NetPhos 3.1 predicted phosphorylation sites by exploiting local sequential patterns in combination with neural network learning. However, different from NetPhos 3.1, PhosContext2vec incorporated both residue-level and contextual-level feature vectors in an integrated manner. In addition, the contextual feature vectors were generated from a number of different contextual patterns present in large protein sequence databases, which could explain why PhosContext2vec achieved a superior performance compared to NetPhos 3.1.

Among the three types of phosphorylation sites, we found that the performance of the S site was better than that of the T site, while the performance of Y site was the worst. According to a previous study [35], PROSITE motifs could only recognize only 10% of annotated Y phosphorylation sites [68], while being able to recognize 48% and 38% of respective S and T phosphorylation sites. This indicates that the local patterns of phosphorylated tyrosine sites are much more difficult to capture, making them more difficult predictions. **Table 7.3** provides the sensitivity, specificity and MCC of the compared predictor for general phosphorylation prediction.

#### **7.4.8 Independent test for kinase-specific phosphorylation site prediction**

In this section, in order to validate the performance of PhosContext2vec for kinase-specific phosphorylation site prediction, we further performed an independent test using the datasets extracted from Swiss-Prot/UniProt and Phospho.ELM. As with general phosphorylation site prediction, 10-fold cross-validation was performed on the SVM models with selected hyper-parameters, from which the model with the best performance on valid set was applied to the independent test set. The compared models include GPS 3.0, MusiteDeep, Musite 1.0, NetPhos 3.1, KinasePhos 2.0, PhosphoPredict and

PhosphoPick. **Figure 7.5** shows the ROC curves and the corresponding AUC scores of these methods. The ROC curves of different methods are indicated by different colors. Five panels correspond to the prediction results of AGC/PKA, AGC/PKC, CMGC/CDK, Other/CK2 and TK/SRC kinase families, respectively. The performance of a method was denoted as “N/A” if such method could not be used to predict phosphorylation sites of a corresponding kinase family.



**Figure 7.5** ROC curves of PhosContext2vec and seven existing methods for kinase-specific phosphorylation site prediction on the independent test.

As shown in **Figure 7.5**, different predictors achieved different performance for the five tested kinases. For all tested kinases, the most recent MusiteDeep method achieved the best performance among all compared methods. For AGC/PKA, Musite 1.0 performed the second-best with an average AUC of 0.956, surpassing the third-best predictor PhosContext2vec by 0.006. Another predictor that achieved an average AUC over 0.9 was GPS 3.0. Compared to the four other predictors that achieved the top AUC scores, KinasePhos 2.0 and PhosphoPick achieved the worst AUC scores between 0.6 and 0.7. For AGC/PKC and CGMC/CDK, PhosContext2vec achieved the second-best AUC scores of 0.939 and 0.960, respectively. Compared to the predictors with the third-best performance, PhosContext2vec increased the AUC value by 0.067 and 0.037, compared to Musite 1.0 and GPS 3.0, respectively. For Other/CK2, PhosphoPick could not predict the potential phosphorylation sites for this kinase family, while all the other predictor achieved the most similar performance with the best AUC score of 0.950 by MusiteDeep and the worst AUC score of 0.840 by PhosphoPredict. Finally, for TK/SRC, GPS 3.0 achieved the best performance with an average AUC score of 0.830, while PhosContext2vec achieved the second-best average AUC score of 0.794. Overall, MusiteDeep, Musite 1.0, GPS 3.0 and PhosContext2vec were evaluated as the top performing predictors among all compared methods, while PhosphoPick and KinasePhos 2.0 performed the worst.

The MusiteDeep is the only method that employed the deep learning architecture, and its superior performance is consistent despite its use of raw protein sequence as the direct input. As an effective contextual feature vector, the PhosContext2vec can be combined as a side channel [352] for further improving the performance of MusiteDeep. The inferior performance of KinasePhos 2.0 may be explained partly by the incomplete results obtained from its web server. The current web server of KinasePhos 2.0 only predicted phosphorylation sites with scores above the given specificity thresholds (four options are available: default, 80%, 90% and 100%). NetPhos 3.1 performed well for

Other/CK2 and TK/SRC and relatively well for other kinase families, for example, AGC/PKA. PhosphoPick achieved an inferior performance for most tested kinases. It is the only predictor developed based on integrating protein functional features such as protein-protein interactions. However, on the other hand, it did not consider any features from amino acid sequences [220]. We would recommend that it should be used in combination with other sequence-based predictors to achieve better performance.

To quantify the performance of different predictors in terms of other measurements, we also calculated sensitivity, specificity, and MCC values. As mentioned in **Section 7.4.3**, the low, medium and high cut-off FPRs for S/T sites were set to 2%, 6% and 10%, while the low, medium and high cut-off FPRs for T sites were set to 4%, 9% and 15%, respectively. In **Table 7.3** and **Table 7.4**, we used the same cut-off FPRs for comparing the different predictors.



**Table 7.3** Performance comparison between different predictors of general phosphorylation sites, evaluated in terms of Sensitivity (SE), Specificity (SP) and Matthews correlation coefficient (MCC).

	Predictor	GPS 3.0			MusiteDeep			Musite 1.0			NetPhos 3.1			PhosContext2vec		
Datasets	Cutoff threshold	SE	SP	MCC	SE	SP	MCC	SE	SP	MCC	SE	SP	MCC	SE	SP	MCC
PPA.S	Low	0.090	0.980	0.082	<b>0.288</b>	0.980	<b>0.280</b>	<u>0.208</u>	0.980	<u>0.206</u>	0.030	0.989	0.029	0.112	0.980	0.106
	Medium	0.255	0.940	0.136	<b>0.452</b>	0.940	<b>0.262</b>	<u>0.385</u>	0.941	<u>0.222</u>	0.120	0.953	0.059	0.271	0.940	0.147
	High	0.366	0.900	0.148	<b>0.532</b>	0.899	<b>0.235</b>	<u>0.468</u>	0.901	<u>0.203</u>	0.221	0.901	0.070	0.375	0.900	0.153
PPA.T	Low	0.055	0.980	0.047	<b>0.124</b>	0.980	<b>0.131</b>	<u>0.097</u>	0.980	<u>0.102</u>	0.026	0.981	0.009	0.077	0.980	0.075
	Medium	0.138	0.940	0.063	<b>0.235</b>	0.940	<b>0.137</b>	<u>0.194</u>	0.941	<u>0.107</u>	0.096	0.941	0.030	0.164	0.940	0.083
	High	0.211	0.901	0.073	<b>0.308</b>	0.900	<b>0.131</b>	<u>0.253</u>	0.901	<u>0.099</u>	0.154	0.901	0.035	0.229	0.898	0.081
PPA.Y	Low	0.043	0.966	0.012	<b>0.095</b>	0.959	<b>0.062</b>	<u>0.089</u>	0.953	<u>0.045</u>	0.074	0.950	0.026	0.077	0.950	0.029
	Medium	0.107	0.913	0.017	<b>0.179</b>	0.909	<b>0.071</b>	<u>0.151</u>	0.909	<u>0.049</u>	0.120	0.911	0.025	0.148	0.908	0.046
	High	0.179	0.850	0.020	<b>0.284</b>	0.850	<b>0.088</b>	0.234	0.850	0.056	0.200	0.850	0.033	<u>0.238</u>	0.850	<u>0.058</u>

\* The best performance is highlighted in bold, while the second-best performance is highlighted with underlines.

**Table 7.4** Performance comparison between different predictors of kinase-specific phosphorylation sites, evaluated in terms of Sensitivity (SE), Specificity (SP) and the Matthews correlation coefficient (MCC).

		GPS 3.0			MusiteDeep			Musite 1.0			NetPhos 3.1		
<i>Datasets</i>	<i>Cutoff threshold</i>	<i>SE</i>	<i>SP</i>	<i>MCC</i>	<i>SE</i>	<i>SP</i>	<i>MCC</i>	<i>SE</i>	<i>SP</i>	<i>MCC</i>	<i>SE</i>	<i>SP</i>	<i>MCC</i>
<b>AGC/PKA</b>	Low	0.464	0.981	0.338	<b>0.867</b>	0.980	<b>0.570</b>	<u>0.641</u>	0.980	<u>0.442</u>	0.503	0.980	0.356
	Medium	0.718	0.940	0.308	<b>0.955</b>	0.939	<b>0.404</b>	<u>0.862</u>	0.941	<u>0.372</u>	0.702	0.940	0.301
	High	0.829	0.900	0.276	<b>0.961</b>	0.900	<b>0.323</b>	0.878	0.900	<u>0.294</u>	0.779	0.900	0.258
<b>AGC/PKC</b>	Low	0.184	0.979	0.154	<b>0.821</b>	0.980	<b>0.618</b>	0.321	0.980	0.274	0.158	0.980	0.135
	Medium	0.372	0.940	0.183	<b>0.893</b>	0.940	<b>0.454</b>	0.536	0.940	0.273	0.316	0.940	0.152
	High	0.577	0.900	0.223	<b>0.929</b>	0.899	<b>0.374</b>	0.673	0.901	0.267	0.429	0.900	0.156
<b>CMGC/CDK</b>	Low	<u>0.423</u>	0.979	<u>0.345</u>	<b>0.901</b>	0.980	<b>0.661</b>	0.493	0.981	<u>0.407</u>	0.338	0.980	0.287
	Medium	<u>0.746</u>	0.938	0.376	<b>1.000</b>	0.939	<b>0.502</b>	0.739	0.941	0.379	0.655	0.940	0.335
	High	<u>0.803</u>	0.900	<u>0.322</u>	<b>1.000</b>	0.899	<b>0.404</b>	0.768	0.901	0.308	0.768	0.900	0.307
<b>Other/CK2</b>	Low	<u>0.493</u>	0.980	<u>0.424</u>	<b>0.735</b>	0.977	<b>0.566</b>	0.493	0.979	0.420	0.390	0.980	0.348
	Medium	<u>0.728</u>	0.940	<u>0.399</u>	<b>0.838</b>	0.940	<b>0.456</b>	0.750	0.941	<u>0.412</u>	0.640	0.940	0.351
	High	0.794	0.900	0.342	<b>0.904</b>	0.894	<b>0.382</b>	0.831	0.901	0.361	0.750	0.901	0.323
<b>TK/SRC</b>	Low	<b>0.289</b>	0.960	<b>0.291</b>	-	-	-	0.274	0.960	0.277	0.200	0.961	0.202
	Medium	<b>0.519</b>	0.912	<b>0.365</b>	-	-	-	0.385	0.912	0.263	0.385	0.911	0.262
	High	<u>0.593</u>	0.854	<u>0.323</u>	-	-	-	0.519	0.851	0.270	0.519	0.851	0.270

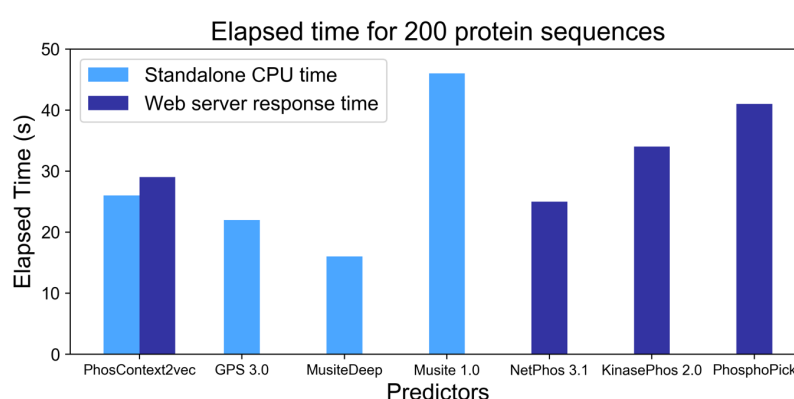
(Continue) Performance comparison between different predictors of kinase-specific phosphorylation sites, evaluated in terms of Sensitivity (SE), Specificity (SP) and the Matthews correlation coefficient (MCC).

		KinasePhos 2.0			PhosphoPredict			PhosphoPick			PhosContext2vec		
Datasets	Cutoff threshold	SE	SP	MCC	SE	SP	MCC	SE	SP	MCC	SE	SP	MCC
AGC/PKA	Low	0.193	0.980	0.139	0.569	0.982	0.412	0.398	0.980	0.286	0.444	0.980	0.318
	Medium	0.320	0.940	0.126	0.608	0.940	0.259	0.459	0.940	0.191	0.783	0.936	0.327
	High	0.403	0.900	0.117	0.724	0.902	0.240	0.475	0.900	0.145	0.850	0.900	0.285
AGC/PKC	Low	0.163	0.979	0.136	0.194	0.980	0.166	0.235	0.980	0.203	<u>0.354</u>	0.980	<u>0.301</u>
	Medium	0.224	0.940	0.099	0.214	0.973	0.157	0.357	0.940	0.175	<u>0.733</u>	0.940	<u>0.374</u>
	High	0.270	0.900	0.081	0.296	0.900	0.094	0.474	0.900	0.176	<u>0.821</u>	0.893	<u>0.319</u>
CMGC/CDK	Low	0.049	0.980	0.030	0.394	0.981	0.334	0.359	0.980	0.302	0.331	0.980	0.282
	Medium	0.092	0.940	0.019	0.542	0.960	0.335	0.472	0.939	0.236	<u>0.824</u>	0.940	<u>0.420</u>
	High	0.148	0.900	0.023	0.542	0.960	0.335	0.507	0.900	0.191	<u>0.915</u>	0.898	<u>0.366</u>
Other/CK2	Low	<u>0.544</u>	0.979	<u>0.454</u>	0.213	0.980	0.198	-	-	-	0.368	0.980	0.329
	Medium	0.721	0.937	0.387	0.397	0.941	0.215	-	-	-	0.654	0.938	0.354
	High	<u>0.868</u>	0.898	<u>0.374</u>	0.449	0.901	0.179	-	-	-	0.765	0.899	0.327
TK/SRC	Low	0.185	0.957	0.174	0.148	0.959	0.137	0.096	0.951	0.059	0.207	0.959	0.204
	Medium	0.252	0.911	0.150	0.237	0.910	0.136	0.148	0.909	0.055	0.304	0.910	0.193
	High	0.326	0.846	0.128	0.274	0.853	0.097	0.237	0.844	0.062	<b>0.637</b>	0.848	<b>0.345</b>

\* The best performance is highlighted in bold, while the second-best performance was highlighted with underlines.

#### 7.4.9 Comparison of different method in terms of time efficiency

To better understand the processing efficiency of different methods, we measured the elapsed time for processing the same 200 protein sequences using each predictor. For example, for general phosphorylation site prediction, we predicted only Y phosphorylation sites. For kinase-specific phosphorylation site prediction, we only predicted phosphorylation sites for the SRC kinase. The five compared predictors included GPS 3.0, MusiteDeep, Musite 1.0, NetPhos 3.1, KinasePhos 2.0 and PhosphoPick. For these predictors, we tested the standalone Java programs of the former two and the online web servers of the latter three. For PhosContext2vec, we tested both the standalone CPU time and the response time of the web server. Most of the existing prediction methods provided either standalone programs or online web servers. Therefore, we calculated the CPU time (denoted by light blue) for those predictors with available standalone programs; and estimated the response time (denoted by dark blue) for those predictors with only web servers available. For PhosContext2vec, both the CPU time and the web server response time were provided.



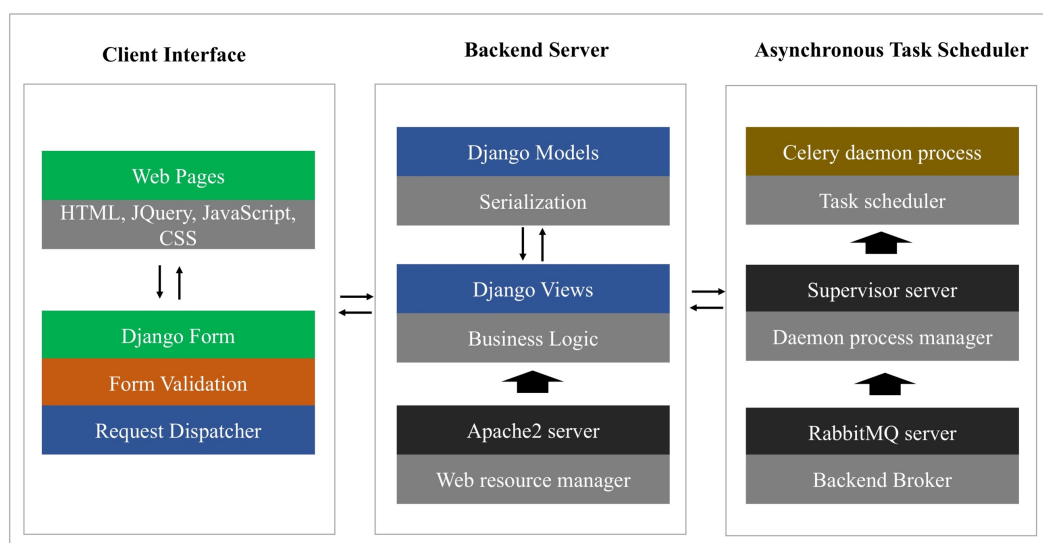
**Figure 7.6** Elapsed time comparison between different phosphorylation site predictors for predicting 200 protein sequences.

As shown in **Figure 7.6**, among the four tools PhosContext2vec, GPS 3.0 MusiteDeep and Musite 1.0 that were evaluated in terms of the CPU time, MusiteDeep executed fastest within 16 seconds. In particular, the CPU time of Musite 1.0 was calculated given that protein disorder information has been cached. Similarly, the CPU time of PhosContext2vec was calculated on the assumption that the BLAST results were cached. It took Musite 1.0 ~45 seconds to process 200 protein sequences, while it took PhosContext2vec about 25 seconds to complete the same prediction task. In terms of the web server response time, NetPhos 3.1 benefited from the use of multiple threads and achieved the fastest speed within 25 seconds. As a comparison, it took KinasePhos 2.0 and PhosphoPick ~35 seconds and more than 40 seconds, respectively, for processing the same 200 sequences. It took the PhosContext2vec web server about 28 seconds to complete the prediction of 200 protein sequences.

#### **7.4.10 Implementation of the PhosContext2vec web server**

The PhosContext2vec web server is currently configured and hosted on a virtual server machine deployed in the Monash e-Research Centre at Monash University, equipped with four cores, 12 GB memory and a 110 GB hard disk. It was implemented using the Python-Django framework [353] and consists of three major components, *i.e.* the client interface, the backend server and the asynchronous task scheduler. The user interface interacts with the clients, collects protein sequence inputs, parameter settings, and email addresses and forwards the submitted requests to the backend server. The backend server interacts with the client interface and determines the logic of each request. Following the submission of each request, the backend server starts a running job and forwards the generated results to the client interface once the job is completed. For long-running jobs, we employed a third component, which is an asynchronous task scheduler, for optimising the allocation of computational resources. With this architecture, the backend server can return a real-time task status before the task is completed, which

sets the server threads free to deal with more requests. **Figure 7.7** shows the architecture of the PhosContext2vec web server.



**Figure 7.7** The architecture of the PhosContext2vec web server.

The PhosContext2vec web server provides three different but complementary functions for users, including contextual feature vector generation, general phosphorylation site prediction and kinase-specific phosphorylation site prediction. Users can submit their protein sequences of interest through either entering the sequence information in the text area provided at the web page, or uploading the sequence file in the FASTA format, or supplying a concise list of Swiss-Prot/UniProt IDs of query proteins. In terms of contextual vector generation, four contextual window sizes (*i.e.* 7, 11, 15, and 19) are available as options. For general and phosphorylation site prediction, the contextual window sizes for different types of phosphorylation sites (including S, T and Y) and kinase families (including AGC/PKA, AGC/PKC, CMGC/CDK, Other/CK2, and TK/SRC) were set to the optimal values based on the empirical results. Users can adjust the prediction cut-off and the output thresholds for generating customised prediction results in order to meet their specific requirements. Potential phosphorylation sites predicted with scores larger than the output threshold will be

included in the prediction output; however, only those with scores larger than the prediction cut-off threshold will be considered as a positive prediction. On task completion, users can download the text-based results, browse the webpage-based result summaries and visualize the graphical statistics. Alternatively, for users who provide their email address, an email with the output links and file attachments will be sent.

#### **7.4.11 Conclusion**

Based on the distributed contextual representation, we applied the contextual feature vector to solve the prediction problems of both general and kinase-specific phosphorylation sites. We conducted cross-validation tests for optimizing the selection of contextual window sizes, contextual representations, and hyper-parameters. Performance improvements were achieved for all the three types of phosphorylation sites and five kinase families when the contextual feature vector was incorporated. When evaluated on the independent test and compared with several state-of-the-art predictors, our method PhosContext2vec achieved superior performance for predicting Y phosphorylation sites, and also for the AGC/PKC and CMGC/CDK kinase families. It also achieved the second-best performance for S and T phosphorylation sites, and for AGC/PKA and TK/SRC kinase families.

As for kinase-specific phosphorylation site prediction, the PhosContext2vec online web server is designed to predict the potential phosphorylation sites of the 138 kinases arranged at different hierarchical levels of kinases, subfamilies and families; however we only performed benchmarking tests for five kinases that had more than 500 experimentally validated phosphorylation sites. In future work, more kinases will be included in the online web server of PhosContext2vec when more experimentally validated phosphorylation data become available for such kinases.

## 7.5 Phosphorylation site prediction based on the DeepS2P framework

### 7.5.1 Introduction

In the previous section, we introduced the application of context2vec to phosphorylation site prediction. The generation of context2vec can be regarded as the process of feature extraction that is decoupled from the SVM-based classification of phosphosites and non-phosphosites. In this section, we introduce the application of the DeepS2P framework to phosphorylation site prediction. The hidden layers in the resulting predictive model, termed DeepS2P-Ph, can be regarded as a feature extractor that is jointly trained with the phosphorylation site classifier.

In this section, to evaluate the prediction performance of DeepS2P-Ph, we perform cross-validation tests to determine the values for hyper-parameters. The prediction performance of the resulting predictive models is compared with existing methods for phosphorylation site prediction in independent test.

### 7.5.2 Experimental design

We used the same training and testing datasets for AGC/PKA, AGC/PKC, CMGC/CDK, Other/CK2 and TK/Src that were extracted from the Phospho.ELM database and the Swiss-Prot/UniProt database, as described in **Section 7.4.3**. Please refer to **Table 7.1** for detailed statistics of the training and testing datasets. Here, AGC/PKA, AGC/PKC, CMGC/CDK, Other/CK2 and TK/Src are all protein kinase families. In order to evaluate the prediction performance of DeepS2P-Ph for protein kinases at different levels of abstraction, we further conducted experiments for kinase group CAMK, kinase family AGC/PKC, kinase subfamily CMGC/CDK/CDK5s and protein kinase Other/PLK/PLK-/PLK1. **Table 7.5** demonstrates the statistics of the datasets for CAMK, AGC/PKC, CMGC/CDK/CDK5s, and Other/PLK/PLK-/PLK1.



**Table 7.5** Statistics of the training and testing datasets for the four kinase nodes representing different levels of abstraction.

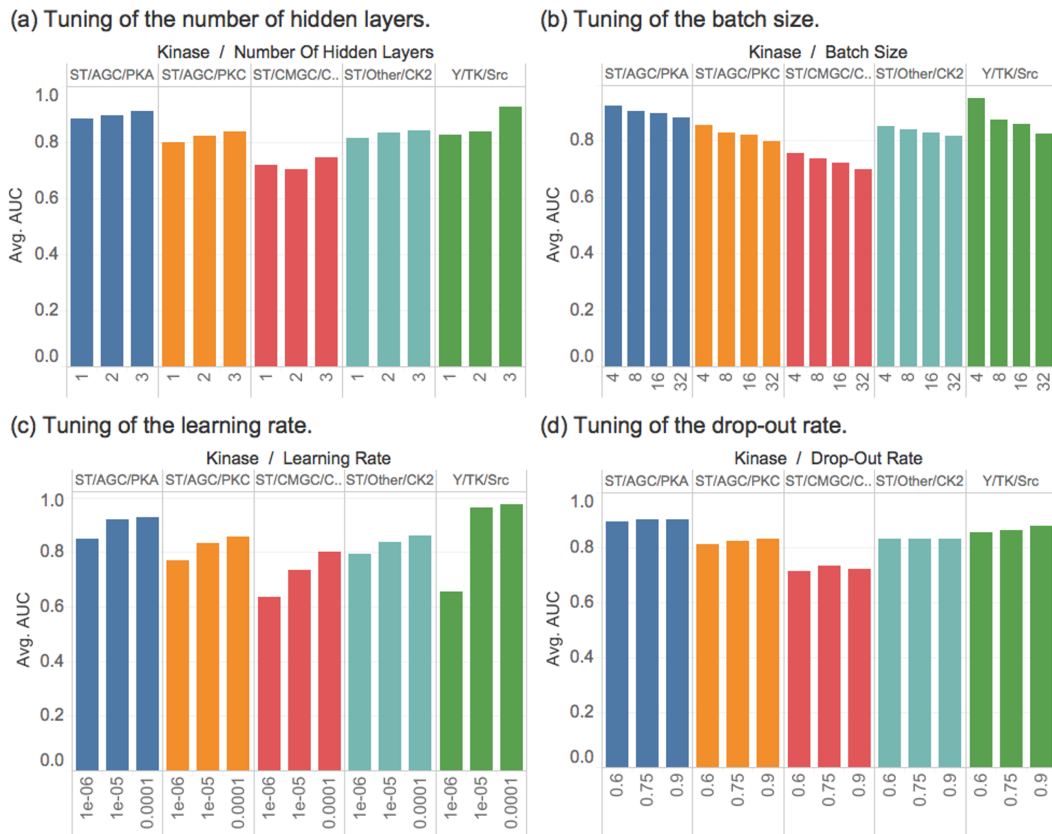
		Training	Testing
CAMK	#pos	651	298
	#neg	48,730	15,744
AGC/PKC	#pos	763	194
	#neg	39,607	12,518
CMGC/CDK/CDK5s	#pos	131	70
	#neg	12,081	4,522
Other/PLK/PLK-/PLK1	#pos	108	30
	#neg	8,932	1,980

We performed 10-fold cross-validation tests on the training datasets and independent tests on the testing datasets. Here, instead of using the negative sampling strategy to ensure a ratio of 1:1 between positive and negative samples we used the full set of non-phosphorylated sites as the negative examples. More specifically, in each epoch of the training process, we extracted a different set of non-phosphorylation sites as the negative examples, keeping a positive to negative ratio of 1:1. In this way, all non-phosphorylated sites were used at least once for training the model, while at the same time keeping the balance between positive and negative examples. Here, non-phosphorylated sites refer to S/T/Y sites that are not annotated as phosphorylation sites. The prediction performance of DeepS2P-Ph was evaluated with the AUC score (see **Appendix A**).

### 7.5.3 Determination of hyper-parameters

According to **Section 4.2.5**, to determine the final structure of the DeepS2P-Ph model, the values of multiple hyper-parameters need to be determined. In this section, we performed cross-validation tests to determine the hyper-parameters of DeepS2P-Ph for each of the five protein kinase families AGC/PKA, AGC/PKC, CMGC/CDK, Other/CK2, and TK/Src. The four hyper-parameters that were tuned for DeepS2P-Ph include the number of hidden layers  $|H|$  ( $|H| \in 1, 2, 3$ ) the batch sizes  $n$  ( $n \in 8, 16, 32, 64$ ), the

learning rate  $\vartheta$  ( $\vartheta \in 1e-4, 1e-5, 1e-6$ ) and the drop-out rate  $\vartheta$  ( $\vartheta \in 0.6, 0.75, 0.9$ ). Therefore, all together, we performed 3x4x3x3 cross-validation tests for each of the five protein kinase families. **Figure 7.8** demonstrates the results of hyper-parameter tuning for four protein families. The results for AGC/PKA, AGC/PKC, CMGC/CDK, Other/CK2, and TK/Src were labeled in color blue, orange, red, turquoise, and green, respectively. The tuning results for the number of hidden layers  $|H|$ , the batch sizes  $n$ , the learning rate  $\vartheta$  and the drop-out rate  $\vartheta$  were demonstrated in subgraph (a), (b), (c) and (d), respectively.

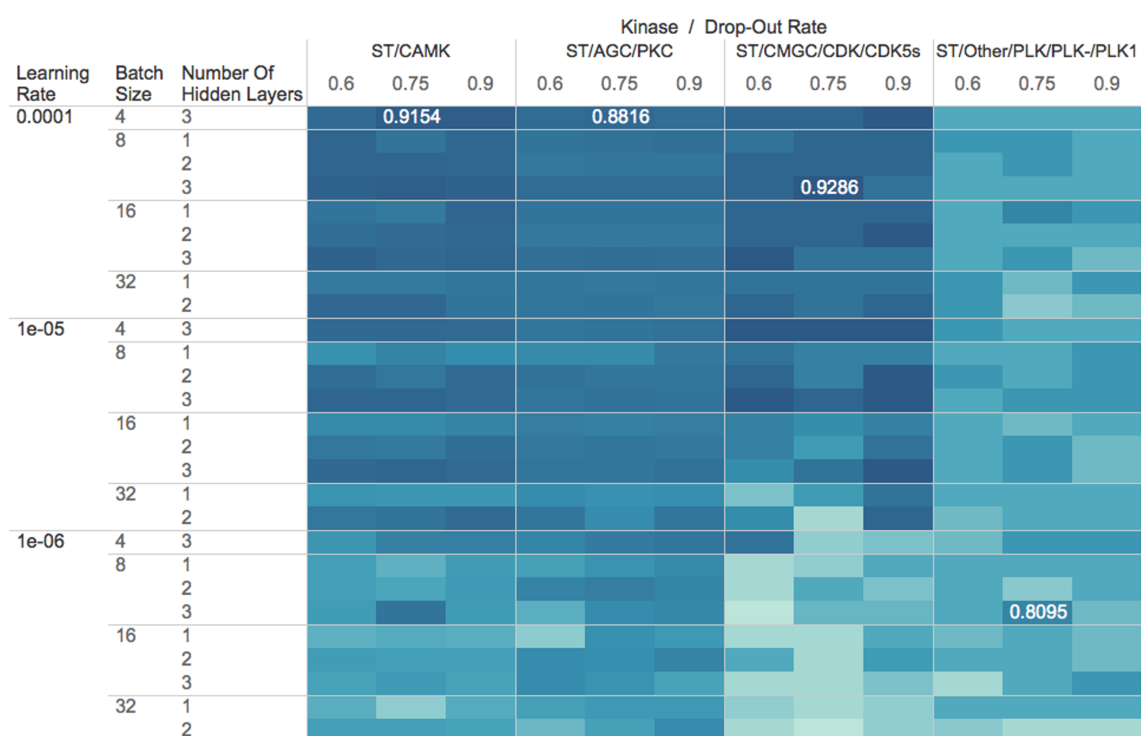


**Figure 7.8** Hyper-parameter tuning for DeepS2P-Ph models in predicting phosphosites of AGC/PKA, AGC/PKC, CMGC/CDK, Other/CK2, and TK/Src.

According to **Figure 7.8**, the prediction performance of DeepS2P-Ph models was optimized with the number of hidden layers set to 3, the batch size set to 4, the learning

rate set to 0.0001 and the (keep probability of the) drop-out rate set to 0.9 for all five tested protein kinase families.

To further determine the hyper-parameters of DeepS2P-Ph for the four kinase nodes, *i.e.* CAMK, AGC/PKC, CMGC/CDK/CDK5s, and Other/PLK/PLK-/PLK1, representing four different levels of abstraction, we performed 10-fold cross-validation tests and plotted the full results as a heat map in **Figure 7.9**, in which the deep blue colour indicates a higher AUC score.



**Figure 7.9** Hyper-parameter tuning for DeepS2P-Ph models in predicting phosphosites of CAMK, AGC/PKC, CMGC/CDK/CDK5s, and Other/PLK/PLK-/PLK1.

According to the results shown in **Figure 7.9**, for CAMK, AGC/PKC, CMGC/CDK/CDK5s, better prediction performance was also achieved with higher learning rates, smaller batch sizes and more hidden layers. For Other/PLK/PLK-/PLK1, the best prediction performance was achieved with the learning rate set to 1e-6, the batch size set to 8, the number of hidden layers set to 3, and the drop-out rate set to

0.75. However, with Other/PLK/PLK-/PLK1, no clear pattern was observed in terms of the preference of hyper-parameters.

Combining the dataset statistics shown in **Table 7.5**, we also observed the correlation between the prediction performance of DeepS2P-Ph and the number of training examples in **Figure 7.9**. In general, DeepS2P-Ph performed better for kinase nodes with more training data. With hyper-parameters optimized, it achieved with an AUC score of 0.8095 for Other/PLK/PLK-/PLK1, which had the least number of training examples (including positive and negative examples). Exceptionally, despite the relatively small training datasets (with 131 positive examples and 12,081 negative examples) for CMGC/CDK/CDK5s, DeepS2P-Ph achieved an AUC score of 0.9286 when all hyper-parameters being optimized.

#### **7.5.4 Independent test in kinase-specific phosphorylation site prediction**

For kinase family AGC/PKA, AGC/PKC, CMGC/CDK, Other/CK2, and TK/Src, we performed independent tests to compare the prediction performance of DeepS2P-Ph with existing phosphorylation site prediction methods on the testing datasets. We also involved the PhosContext2vec method that was introduced in **Section 7.4**. Please refer to **Section 2.3.3** for detailed introduction of the compared prediction methods.

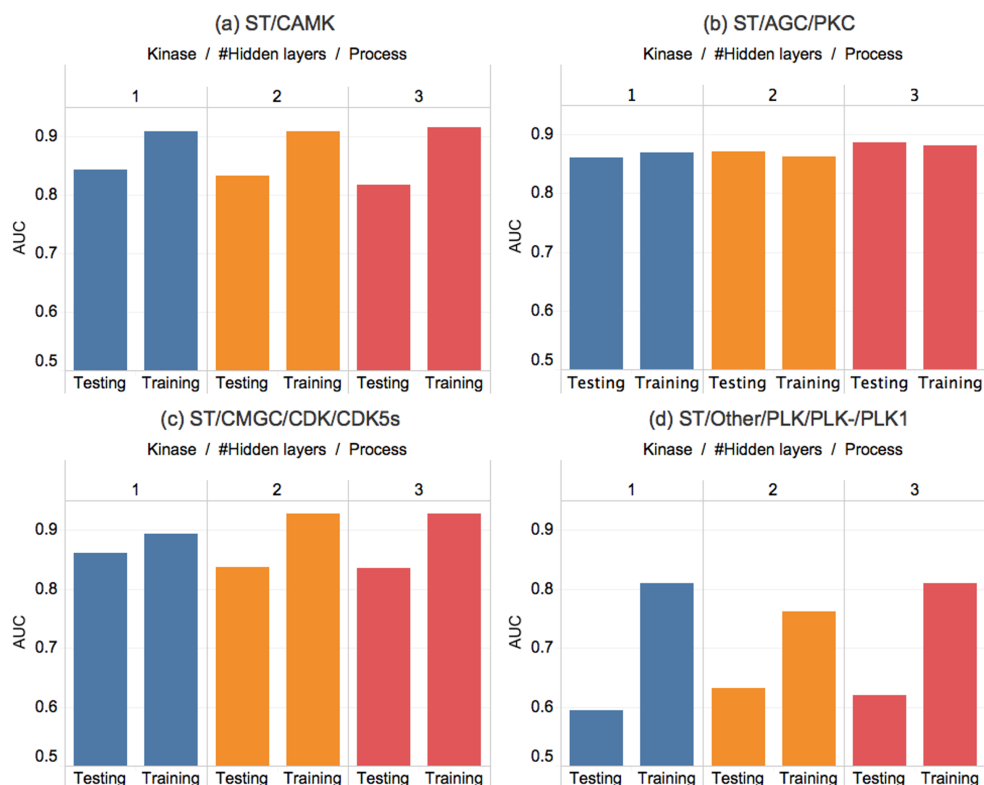
**Table 7.6** Prediction performance comparison between DeepS2P-Ph and existing phosphorylation site prediction methods in terms of AUC scores.

	AGC/PKA	AGC/PKC	CMGC/CDK	Other/CK2	TK/Src
GPS	0.934	0.845	0.923	0.935	<b>0.830</b>
Musite	0.956	0.872	0.898	0.922	0.749
MusiteDeep	<b>0.976</b>	<b>0.954</b>	<b>0.973</b>	<b>0.950</b>	n/a
NetPhos	0.897	0.754	0.861	0.892	0.775
KinasePhos	0.673	0.621	0.511	0.915	0.570
PhosphoPredict	0.911	0.733	0.849	0.840	0.658
PhosphoPick	0.619	0.679	0.657	N/A	0.540
PhosContext2vec	0.950	0.939	0.960	0.912	0.794
DeepS2P-Ph	0.899	0.886	0.838	0.822	0.732

\* The best performance in terms of AUC score for each kinase was highlighted as bold.

According to the results in **Table 7.6**, we observed that the DeepS2P-Ph method outperformed the KinasePhos method for all tested kinase families except for Other/CK2, outperformed PhosphoPick for kinase family AGC/PKA, AGC/PKC, CGMC/CDK, and TK/Src, and also outperformed PhosphoPredict for kinase family AGC/PKC and TK/Src. However, it achieved less favourable performance compared to the state-of-the-art methods such as GPS, MusiteDeep, Musite and PhosContext2vec. It also achieved the worst AUC scores for kinase family Other/CK2.

Considering the number of the training examples of the five protein kinase families, we performed additional independent tests to investigate whether the unsatisfying prediction performance was caused by overfitting. Since overfitting is directly related to the number of training examples, we applied the DeepS2P-Ph model to kinase group CAMK, kinase family AGC/PKC, kinase subfamily CGMC/CDK/CDK5s and protein kinase Other/PLK/PLKs/PLK1, with a data set of 49,831, 40,370, 12,212 and 9,040 training examples, respectively. By investigating the relation between the number of training examples and the difference between the training performance and testing performance, we investigated whether the overfitting is caused by a lack of training data and whether the unsatisfying prediction performance is caused by overfitting. Note that overfitting is often indicated by strong performance during the training process and weak performance in independent tests.



**Figure 7.10** Prediction performance comparison of DeepS2P-Ph in cross-validation (training) and independent (testing) tests.

According to the results shown in **Figure 7.10**, for AGC/PKC, the prediction performance of DeepS2P-Ph demonstrated the least difference between that achieved in the training process and that achieved in the testing process. In contrast, for Other/PLK/PLK-/PLK1, the performance of DeepS2P-Ph achieved in the testing process was the most different from that achieved in the training process. More specifically, the AUC scores dropped by 0.215, 0.130, 0.190 in independent tests compared to those achieved in cross-validation tests. It is noteworthy that, among the four kinase nodes, AGC/PKC and Other/PLK/PLK-/PLK1 respectively had the most and least number of training examples, which represents one of the direct factors related to overfitting. Therefore, we can conclude that the DeepS2P-Ph model has a tendency towards overfitting when there is a lack of training data.

From the results of CAMK and CMGC/CDK/CDK5s, we also observed the increase of the difference between the AUC scores in training and testing process, when the number of hidden layers was increased. For CAMK, as an example the AUC scores were decreased by 0.066, 0.076 and 0.099 in independent test compared to those achieved in the training process, when the number of hidden layers increased from 1 to 3. This pattern of performance change demonstrated that overfitting is more likely to happen when more hidden layers were involved, which indicated the disadvantage of the deep architecture when there is insufficient training data.

### **7.5.5 Conclusion**

In this section, we applied the deep learning framework DeepS2P to kinase-specific phosphorylation site prediction, resulting in the DeepS2P-Ph predictive model. We performed cross-validation tests to determine the values of hyper-parameters and independent tests to evaluate the prediction performance of DeepS2P-Ph. The unsatisfying results achieved by DeepS2P-Ph demonstrated the disadvantage of using DeepS2P framework for prediction tasks with insufficient training data. In order to address this issue, in the next section, we introduce the application of DeepTransfer to predicting phosphosites of kinases with limited number of training examples.

## **7.6 Phosphorylation site prediction using DeepTransfer**

### **7.6.1 Introduction**

It is rather challenging to structurally characterize kinases for the purpose of kinase-specific phosphorylation site prediction. Yet, kinases have been readily classified according to their catalytic domain sequences into groups, families and subfamilies [300]. It was in GPS 2.0 that, for the first time, the hierarchical kinase classification tree (HKCT) was used as heuristic knowledge for designing phosphorylation site prediction models [214]. This also provided a new hierarchical perspective for investigating kinase-



specific phosphorylation. However, due to the lack of phosphosites annotated for specific kinases, few methods approached kinase-specific phosphorylation site prediction in such a hierarchical manner. The more recent MusiteDeep [78] implemented the idea of transfer learning to reuse models trained in general phosphorylation site prediction as the feature extractor for models in kinase-specific phosphorylation site prediction. However, it did not explore the effect of deep learning in the hierarchy of the HKCT, where the hierarchical relations between kinase groups, families, subfamilies and protein kinases can be used as the heuristic for constructing the deep learning framework.

In this section, we introduce the application of the framework DeepTransfer for kinase-specific phosphorylation site prediction, termed *PhosTransfer*. By using deep transfer learning, our prediction models for kinases with limited annotated phosphosites achieved improved performance by reusing the hidden layers of models learned for kinase groups/families/subfamilies with more extensive phosphosite annotations. We further analysed the factors that affect the prediction performance of PhosTransfer, visualized the vector representations generated by hidden layers pre-trained at different tree level, and compared its performance to that of state-of-the-art phosphorylation site prediction methods.

### **7.6.2 The application of DeepTransfer to phosphorylation site prediction**

According to the HKCT [300], there are 8 major kinase groups, each of which has multiple kinase families and subfamilies. Individual kinases were clustered in accordance with this multi-level classification system, forming an HKCT of four levels including kinase *groups*, *families*, *subfamilies* and protein kinases (*PKs*). For simplicity, in what follows, we refer to any tree node including kinase groups, families, subfamilies and PKs in the HKCT as a *kinase node*. Based on this hierarchical tree structure, Xue, Y. et al. (2008) proposed using the annotated phosphosites of kinase nodes at lower levels for training the models of kinase nodes at higher levels. In PhosTransfer, the hierarchical relationships among kinase groups, families, subfamilies and PKs are used to improve

the prediction performance of individual kinase nodes. In this case, the training data is reused in a bottom-up manner.

In general, PhosTransfer refers to a set of predictive models for kinase nodes in HKCT that are organised in a hierarchical structure. The kinase node -specific predictive models in PhosTransfer are constructed with different number of hidden layers depending on the level of the kinase node in HKCT. The predictive models for kinase groups, families, subfamilies, and protein kinases are described as follows.

**Level 1 (Kinase group).** For each of the eight kinase groups introduced by [300], we trained a CNN with their respective training data sets. Each of this CNN had a single hidden layer  $h_1^g$  where  $g \in (\text{AGC, Atypical, CAMK, CK1, CMGC, Other, STE, and TK})$ .

**Level 2 (Kinase family).** For any kinase family  $f$  in group  $g$ , we reused the hidden layer  $h_1^g$  to extract the abstract representation of the input feature vector, based on which the second hidden layer  $h_2^f$  was added and trained to predict phosphorylation sites that were catalysed by kinase family  $f$ . For example, when  $g = \text{AGC}$  kinase family  $f \in (\text{DMPK, GRK, NDK, PKA, PKB, PKC, PKG, ...})$ .

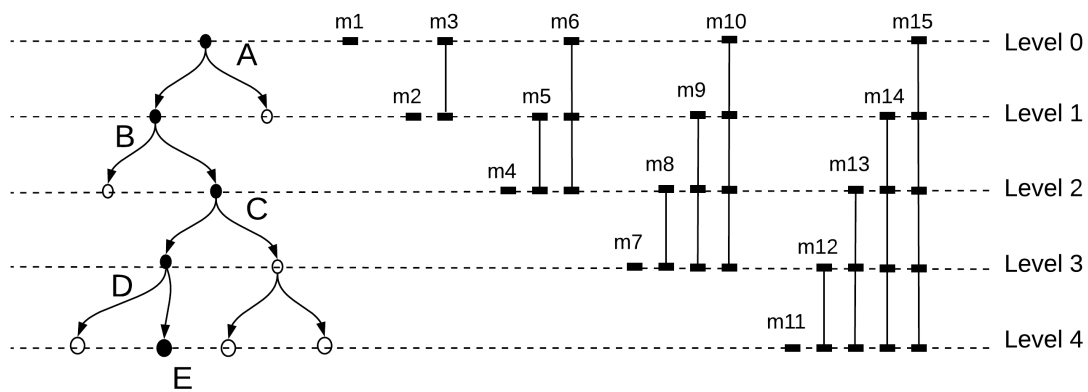
**Level 3 (Kinase subfamily).** For any subfamily  $s$  in kinase family  $f$ , we reused the hidden layer  $h_1^g$  and  $h_2^f$  to extract the abstract representation of input  $x$ , based on which the third hidden layer  $h_3^s$  was added and trained to predict phosphorylation sites that were catalyzed by kinase subfamily  $s$ . For example, when  $g = \text{AGC}$  and  $f = \text{PKC}$  subfamily  $s \in (\text{Alpha, ...})$

**Level 4 (Protein kinase).** For PK  $k$  in kinase subfamily  $s$ , we reused the hidden layer  $h_1^g$ ,  $h_2^f$  and  $h_3^s$  to extract the abstract representation of input  $x$ , based on which the fourth hidden layer  $h_4^k$  was added and trained to predict phosphorylation sites that were catalysed by protein kinase  $k$ . For instance, when  $g = \text{AGC}$ ,  $f = \text{PKC}$  and  $s = \text{Alpha}$ , protein kinase  $k \in (\text{PKC}\alpha, \text{PKC}\beta)$ .

In order to explore the large amount of S/T/Y phosphorylation sites that are not specifically annotated for any kinase, we added an extra level on top of the level group

in the hierarchical tree of kinases, namely, the **Level 0 (Amino-acid type)**, for which the single hidden layer  $h_0^{aa}$  is inserted and trained as the feature extractor for subsequent hidden layers (convolutional filters)  $h_1^g$ ,  $h_2^f$ ,  $h_3^s$  and  $h_4^k$ . Here, the amino acid type  $AA \in (S/T, Y)$ . Among the 8 kinase groups, only the model of TK is trained based on  $h_0^Y$  while the models of other groups are trained based on  $h_0^{S/T}$ .

### 7.6.3 Predictive models for kinase nodes in hierarchy



**Figure 7.11** Derived models for kinase nodes in hierarchies of the HKCT.

**Figure 7.11** demonstrates a partial tree surrounding the tree path A-B-C-D-E, where A, B, C, D and E represent the site type (S/T, or Y), the kinase group, the kinase family, the kinase subfamily and the protein kinase, respectively. For each of the models m3, m5 ~ m6, m8 ~ m10, and m12 ~ m15, there is more than one hidden layer, and the aforementioned deep transfer learning was applied. For example, for model m13, the first hidden layer  $h_2^f$  was trained using annotated phosphosites of family C (effectively model m4), based on which the second hidden layer  $h_3^s$  was trained using annotated phosphosites of subfamily D (effectively model m8). Based on these two pre-trained hidden layers, the third hidden layer  $h_4^k$  was trained using the annotated phosphosites of protein kinase E. Therefore, the objective function [Eq. 4.20] is represented as  $y_4 =$

$\sigma(W_{4o}h_4^k(h_3^s(h_2^f(x))) + b_{4o}); \theta_4$ . Compared to model m15, which starts from **Level 0** and has five hidden layers, model m13 starts from **Level 2**. Therefore, it only has three hidden layers.

Among the 15 compared models, the last hidden layers of models m11 ~ m15 were trained using the phosphosites of protein kinase E, which we refer to as the direct models of E. Similarly, model m1, m2 ~ m3, m4 ~ m6 and m7 ~ m10 are the direct models of aa type A, group B, family C and subfamily D, respectively. According to **Section 4.4.5**, the models of ancestors can also be applied to predict phosphorylation sites of protein kinase E. Therefore, model m1 ~ m10 are referred to as indirect models of protein kinase E.

#### 7.6.4 Experimental design

**Datasets.** We extracted the same set of triple-record annotations for (protein, position, kinase type) from the Swiss-Prot/UniProt database and the Phospho.ELM database, according to the procedures of dataset construction that was described in **Section 7.4.3**. The only difference is that we removed kinases that had less than 15 triple-record annotations instead of less than 20 triple-record annotations. As a result, we obtained consolidated phosphorylation sites for 8 kinase groups, 50 kinase families, 52 kinase subfamilies and 69 kinase types. Please refer to **Appendix A** for the HKCT of kinases for which we constructed training and independent datasets. Finally, we constructed the training and independent test sets for each of the 179 groups/families/subfamilies/PKs by randomly partitioning the datasets using a size ratio of 4:1.

With the above procedure, we constructed the training and testing datasets for kinases classified in four hierarchically organized levels (from top to bottom: groups, families, subfamilies, PKs), with higher levels representing more abstract concepts. To fully explore the annotated phosphorylation sites, we added an extra amino-acid (AA) level on top of the group level, indicating whether the phosphorylation happens on target

residue(s) S/T or Y. When constructing the training and testing datasets, layer-by-layer, we ensured a consistent separation between the training sets and the independent test sets for kinase nodes in the same tree path. More specifically, we excluded the proteins in the test set of protein E from the training set of AA type A, group B, family C, and subfamily D. Therefore, the models trained based on A, B, C, and D do not unfairly favour the test set of E during independent test.

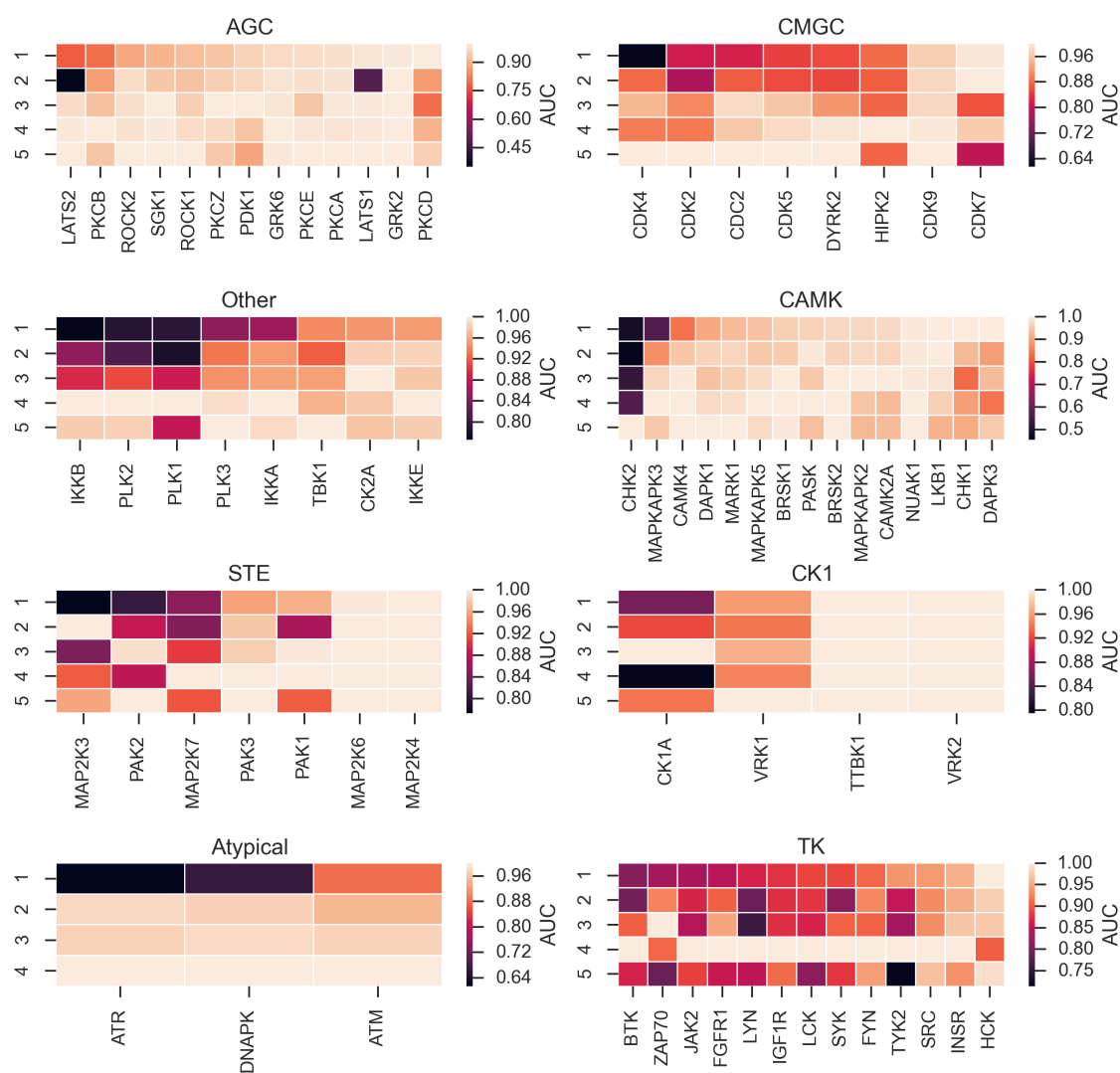
**Performance evaluation.** Similar to the methodology that was introduced in **Section 7.4.3**, we evaluated the prediction performance of PhosTransfer using three measurement scores including the area under the ROC curve (AUC), the Matthew's coefficients of correlation (MCC) and the balanced accuracy (BACC) (see **Appendix A**). BACC measures the balanced accuracy by leveraging the score using the number of positive and negative examples, MCC provides a balanced evaluation of the quality of binary classification, and AUC generates an overview of the ratio between sensitivity and specificity. The former two rely on the selection of the cut-off threshold to distinguish positive predictions from negative ones.

#### 7.6.5 Hyper-parameter tuning

Hyper-parameters for the deep transfer learning framework includes batch size  $n$ , epoch size  $m$ , dropout rate  $d$ , learning rate  $r$ , weight parameter  $\mathbf{W}$ , and bias parameter  $\mathbf{b}$ . We tuned the hyper-parameters for each kinase group, family, subfamily and protein kinase with the different range of values, depending on the level of the kinase node. For Level 0, we tuned the model with  $n \in (128, 64, 32)$  and  $r \in (1e-3, 1e-4)$ . For Level 1 and 2, we tuned the models with  $n \in (32, 16, 8)$  and  $r \in (1e-3, 1e-4)$ . And for Level 3 and 4, we tuned the models with  $n \in (16, 8, 4)$  and  $r \in (1e-4, 1e-5)$ . For all levels, we tuned the models with  $m = 800$  and  $d \in (0.6, 0.75, 0.9)$ . For initialization of  $\mathbf{W}$  and  $\mathbf{b}$ , we drew the initial weight values from a normal distribution and set the initial bias vectors with all elementary values set to 0.1.

### 7.6.6 Results for protein kinases in different groups

We first compared the prediction performance of direct models m11 ~ m15 for the 69 PKs in different groups, in terms of AUC scores. Heat maps in **Figure 7.12** correspond to the 8 tested kinase groups, where models with better performance are highlighted with lighter colour. The results of the 5 models for each PK were normalised considering the difference between the prediction performance for different kinases. Since the five direct models m11, m12, m13, m14, and m15 were constructed with 1, 2, 3, 4, and 5 hidden layers respectively, this investigation helped to visualise the changes in the patterns of relative performances with respect to different number of hidden layers.



**Figure 7.12** Prediction performance of direct models m11 ~ m15 with 1~5 hidden layers for PKs, evaluated using AUC scores.

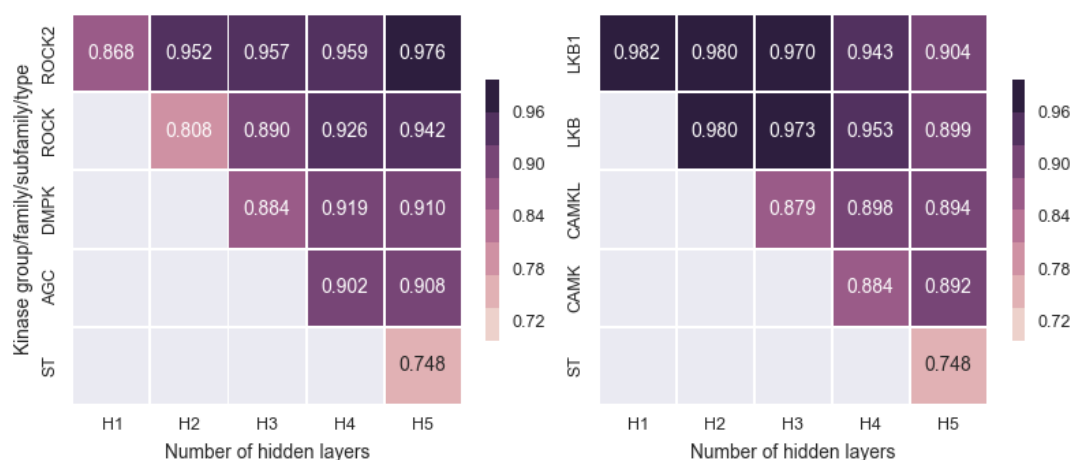
The results in **Figure 7.12** showed that for most tested PKs in kinase groups AGC, CMGC and Atypical, prediction performance was improved with the increase of hidden layers. For models of these PKs, hidden layers pre-trained with the annotated phosphosites of their subfamilies, families, groups and S/T sites were reused as convolutional feature extractors and played a positive role in improving the prediction performance. For most PKs in the group Other, the best performance was achieved by model m14, where the first 3 hidden layers were pre-trained with annotated phosphosites of their groups, families, and subfamilies, respectively. For these PKs, the performance of model m15, which included  $h_0^{aa}$  ( $aa=S/T$ ) as the first hidden layer, was inferior compared to the performance of model m14. This indicates that the phosphosites of S/T sites negatively affected the prediction performance. Considering that the PKs in group Other are structurally different from PKs in other groups, it makes sense that the annotated S/T sites among which most were from other groups did not help in improving the performance.

In group CAMK, better performance was achieved by one of model m12 ~ m15 for most PKs, which demonstrated the positive effect of deep transfer learning. However, for kinases LKB1, CHK1 and DAPK3, the best performance was achieved by model m11 that was trained solely on the annotated phosphosites of PKs themselves. Especially for kinase LKB1, the prediction performance decreased with the increase of the number of hidden layers. It indicates that phosphorylation sites of LKB1 are well distinguished from phosphosites of others, for which further experimental validation are required. Finally, in groups STE and CK1, better performance was generally achieved by one of model m12 ~ m15, but a clear pattern was not observable. Kinase MAP2K3, PAK2 and MAP2K7, as an example, had their best performance achieved with 2, 5, and 4 hidden layers respectively. Only for kinase PAK3 did the prediction performance increase with the number of hidden layers.

For most PKs in kinase group TK, the best prediction performance in terms of AUC scores was achieved by models m14. According to the normalized results, the pre-trained layer  $h_3^s$  in model m12 and the pre-trained layer  $h_2^f$  in model m13 based on phosphosites of subfamilies and families, respectively, played little positive effect in the improvement of the performance. The prediction performance was only improved when an extra layer  $h_1^g$  trained on phosphosites of the group TK was added in model m14 as the first convolutional feature extractor. At the same time, the first layer trained on Y phosphosites, *i.e.*  $h_0^{aa}$  ( $aa=Y$ ), in model m15 played a negative effect in improving the performance, which can be explained by the diverse local sequential patterns of Y phosphosites [35].

### 7.6.7 Results for groups, families and subfamilies

We further evaluated direct models of kinase groups, families, subfamilies and PKs that are on the same paths in the HKCT. **Figure 7.13** demonstrates prediction performances of direct models of all kinase nodes in the paths AGC-DMPK-ROCK-ROCK2 and CAMK-CAMKL-LBK-LBK1, representing two situations where the best performance was achieved with more hidden layers and with less hidden layers, respectively.



**Figure 7.13** Prediction performance of models for groups, families, subfamilies and PKs in two paths.



For kinase ROCK2, improved model performance can be anticipated from the improved performance of their ancestors in the HKCT. The prediction performance of the ROCK2 associated model improved from 0.868 to 0.952, 0.957, 0.959 and 0.976, respectively, when the hidden layers trained on phosphosites of subfamily ROCK, family DMPK, group AGC and amino acid type S/T were added incrementally as the first convolutional feature extractors. The same pattern of incrementally improved performance was also observed for subfamily ROCK (from 0.806 to 0.942), for family DMPK (from 0.884 to 0.910), and for group AGC (from 0.902 to 0.908). The performance of models trained on S/T phosphosite data (0.748 for both paths) was achieved with one hidden layer  $h_0^{aa}$  (aa=S/T). The relatively less satisfying performance of models trained on phosphorylation data for S/T sites is likely due to the greater diversity of substrates for kinases from different groups, families and subfamilies. Similar patterns of positively correlated performance among kinase nodes were observed in most tree paths, with the best performance achieved by deep transfer learning models that used more than one hidden layer.

For path ST-CAMK-CAMKL-LKB-LKB1, the prediction performance of models for different tree nodes demonstrated an opposite pattern of dependency compared to that of the path ST-AGC-DMPK-ROCK-ROCK2. More specifically, the best prediction performance for LKB1 (0.982) was achieved by the model with only one hidden layer, which decreased to 0.970, 0.943, and 0.904 when the layers pre-trained (with phosphosite data for family CAMKL, group CAMK, and type S/T, respectively) were reused for feature extraction. In subfamily LKB, no annotated phosphosites for other PK members were extracted. Therefore, the performance of models for LKB was similar to those for LKB1. However, the performance of models for family CAMKL and group CAMK improved when the number of hidden layers was increased. It indicates that this is a pattern specific to LKB1, not necessarily affecting other PKs in group CAMK. In fact,

according to the plot of group CAMK **Figure 7.12**, prediction performance was improved in 12 out of 15 PKs by applying deep transfer learning.

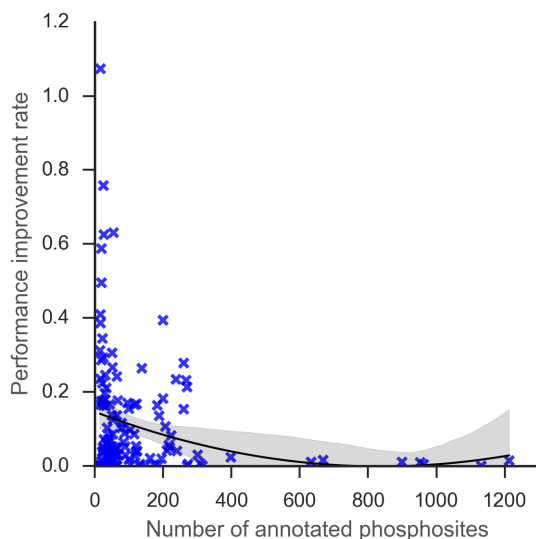
### 7.6.8 Performance improvement for kinases with insufficient annotations

Transfer learning was designed in part to improve the performance for prediction tasks with insufficient training data. Among all the tested 180 kinase groups, families, subfamilies and PKs, ~84% had no more than 200 annotated phosphosites, ~45% had no more than 50 annotated phosphosites, and ~10% had no more than 20 annotated phosphosites. Therefore, we investigated the performance improvements due to deep transfer learning with respect to the number of annotated phosphosites for each kinase node. Given the prediction performance of various kinase, nodes were different even without deep transfer learning, we defined the performance improvement rate (*PIR*): Let *M* and *M'* denote the model with and without deep transfer learning respectively, the measurement *PIR* is defined as

$$PIR(M, M') = \frac{s_M - s_{M'}}{s_{M'}} \quad [\text{Eq. 7.3}]$$

where  $s_M$  and  $s_{M'}$  represent the prediction performance of model *M* and *M'*, respectively. Here, the AUC was used as the performance evaluation score *s*.

According to **Figure 7.11**, models m1, m4, m7 and m11 are based on the single-layer feedforward neural network (SLFN), to which no deep transfer learning was applied. All other models are deep transfer learning models with different numbers of pre-trained hidden layers. Therefore, models m1, m4, m7 and m11 correspond to model *M'* while all others correspond to model *M*. For each tested kinase node *n*, we then calculated the *PIR* between the best performing deep transfer learning model and the associated SLFN-based model and plot the calculated *PIR* score with respect to the number of extracted annotated phosphosites of *n*. **Figure 7.14** depicts the scatter plot, with each point representing a node's relationship between performance improvement rate and the number of annotated phosphosites.



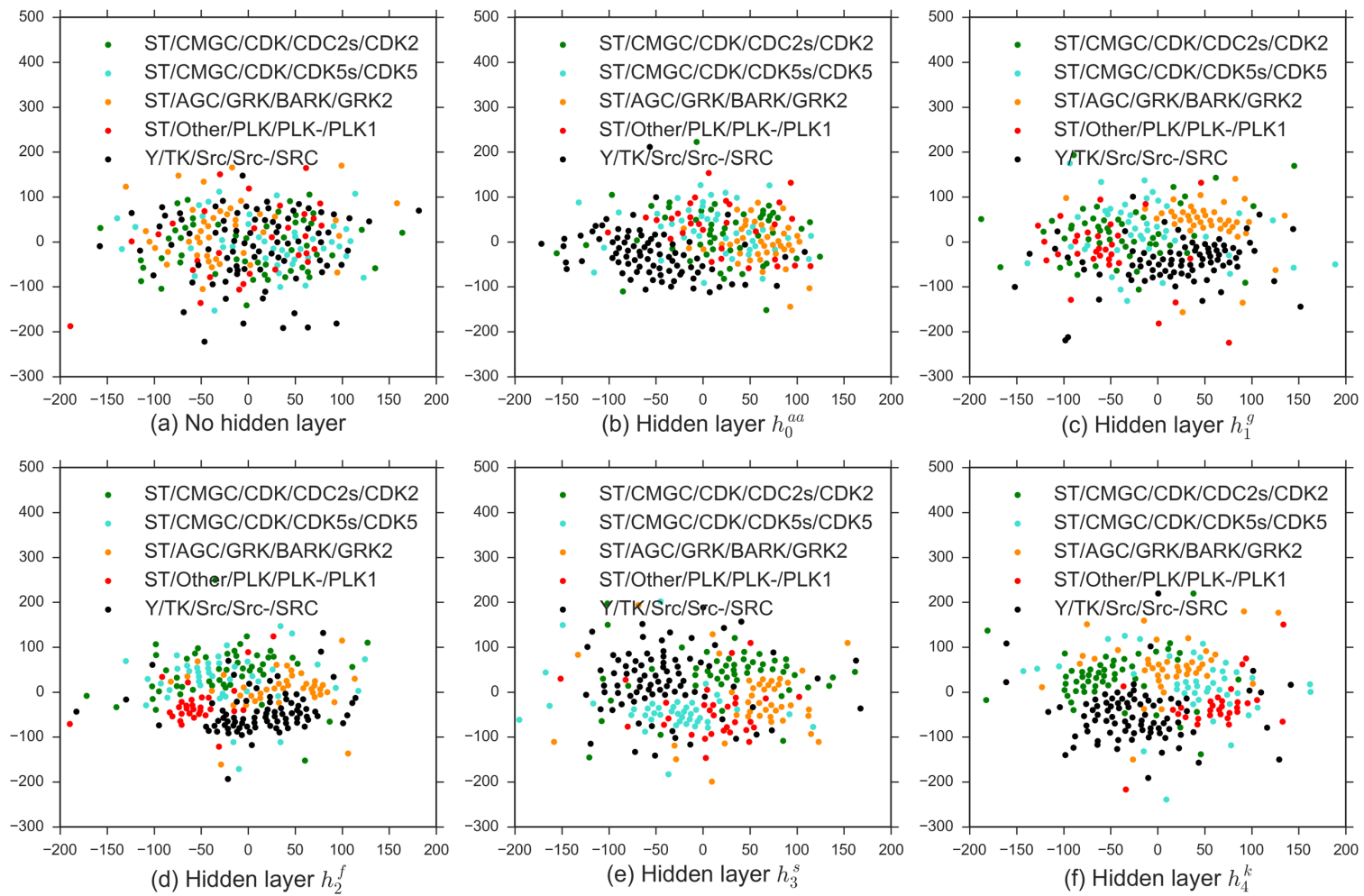
**Figure 7.14** The correlation between the *PIR* and the number of annotated phosphosites.

According to the results, the prediction performance for kinases with insufficient annotated sites is more likely to improve when deep transfer learning was applied. While for kinases with more than 400 annotated sites, the prediction performance was improved by no more than 10% compared to the performance of the SLFN-based models. However, there is no guarantee for kinases with less annotated sites to have their prediction model's performance improved when deep transfer learning is applied. Other factors, such as the local sequential patterns, may also affect the *PIR* of deep transfer learning models. Nevertheless, the results confirmed the positive effect of deep transfer learning in predicting phosphorylation sites for kinases with insufficient annotations.

### 7.6.9 Vector space analysis for features generated by hidden layers

In PhosTransfer, hidden layers pre-trained based on phosphosites of kinase nodes in higher levels of the HKCT are used as the feature extractors for models of kinase nodes in lower levels. Therefore, in model m15 the first hidden layer  $h_0^{aa}$  was pre-trained in model m1, the second hidden layer  $h_1^g$  was pre-trained in model m3, the third hidden

layer  $h_2^f$  was pre-trained in model m6, and fourth hidden layer  $h_3^s$  was pre-trained in model m10. In order to evaluate these hidden layers as the feature extractor, we generated the vector representations from  $h_0^{aa}$ ,  $h_1^g$ ,  $h_2^f$ ,  $h_3^s$  and  $h_4^k$  in model m1, m3, m6, m10 and m15, respectively, for phosphosites of five PKs including CDK2, CDK5, GRK2, PLK1 and SRC. **Figure 7.15** demonstrates the scattered plot of the generated vector representations generated by different hidden layers (feature extractors), where each vector representation was mapped to a 2-dimensional feature vector by using t-SNE.



**Figure 7.15** The t-SNE plot of vector representations generated from hidden layers.

In **Figure 7.15** (a), no hidden layer was used as the feature vector, and the distribution of phosphosites of all five kinases overlap with each other. In **Figure 7.15** (b), hidden layers pre-trained at site level was applied, and the phosphosites of SRC was clustered to the left side of the vector space, while the distributions of phosphosites of other four kinases still overlap. Here, SRC is the only PK that catalyses the Y sites. It indicates that the hidden layer  $h_0^{ST}$  trained with S/T sites and the hidden layer  $h_0^Y$  trained with Y sites are capable of distinguishing phosphosites of kinases catalyse S/T sites and Y sites respectively. In **Figure 7.15** (c), the hidden layers were trained at the group level, and the phosphosites of GRK2, PLK1, and CDK2/CDK5 were separated from each other. Here, GRK2, PLK1 and CDK2/CDK5 belong to group AGC, Other and CMGC, respectively. It indicates that the hidden layers pre-trained at the group level are capable of distinguishing phosphosites of kinases from different groups. In **Figure 7.15** (d), the distribution of phosphosites of CDK2 and CDK5 still overlap with each other, which is consistent with the HKCT where both CDK2 and CDK5 belong to the same kinase family CDK. In **Figure 7.15** (e), the distribution of phosphosites of CDK2 and CDK5 were separated from each other, which validated the classification system where CDK2 and CDK5 were classified to have different subfamily CDK2s and CDK5s, respectively. Finally, in **Figure 7.15** (f), phosphosites of all five kinases were clustered into five groups, which corresponds to the five kinases, respectively.

#### 7.6.10 Comparison with state-of-the-art methods

We first compared the performance of PhosTransfer models to that of the method GPS 3.0. Both PhosTransfer and GPS 3.0 predicted kinase-specific phosphorylation sites according to the hierarchy of kinases, while we used the same HKCT as in the GPS methods. Therefore, most kinases that are available in PhosTransfer are also available in GPS 3.0. To provide an overview of the comparison results, we counted the number of kinase groups, families, subfamilies and PKs for which PhosTransfer and GPS 3.0 performed better, respectively (**Table 7.7**).

**Table 7.7** Prediction performance comparison between PhosTransfer and GPS 3.0.

<i>Group names</i>	# groups		# families		# subfamilies		# PKs	
	<i>M</i>	<i>M'</i>	<i>M</i>	<i>M'</i>	<i>M</i>	<i>M'</i>	<i>M</i>	<i>M'</i>
AGC	1	0	4	4	3	4	7	5
Atypical	0	1	0	1	0	3	--	--
CAMK	1	0	4	4	3	5	4	8
CK1	0	1	1	1	--	--	2	1
CMGC	0	1	2	2	5	5	3	4
STE	0	1	0	3	0	2	2	3
Other	1	0	3	2	--	--	7	1
TK	0	1	1	12	0	0	2	11

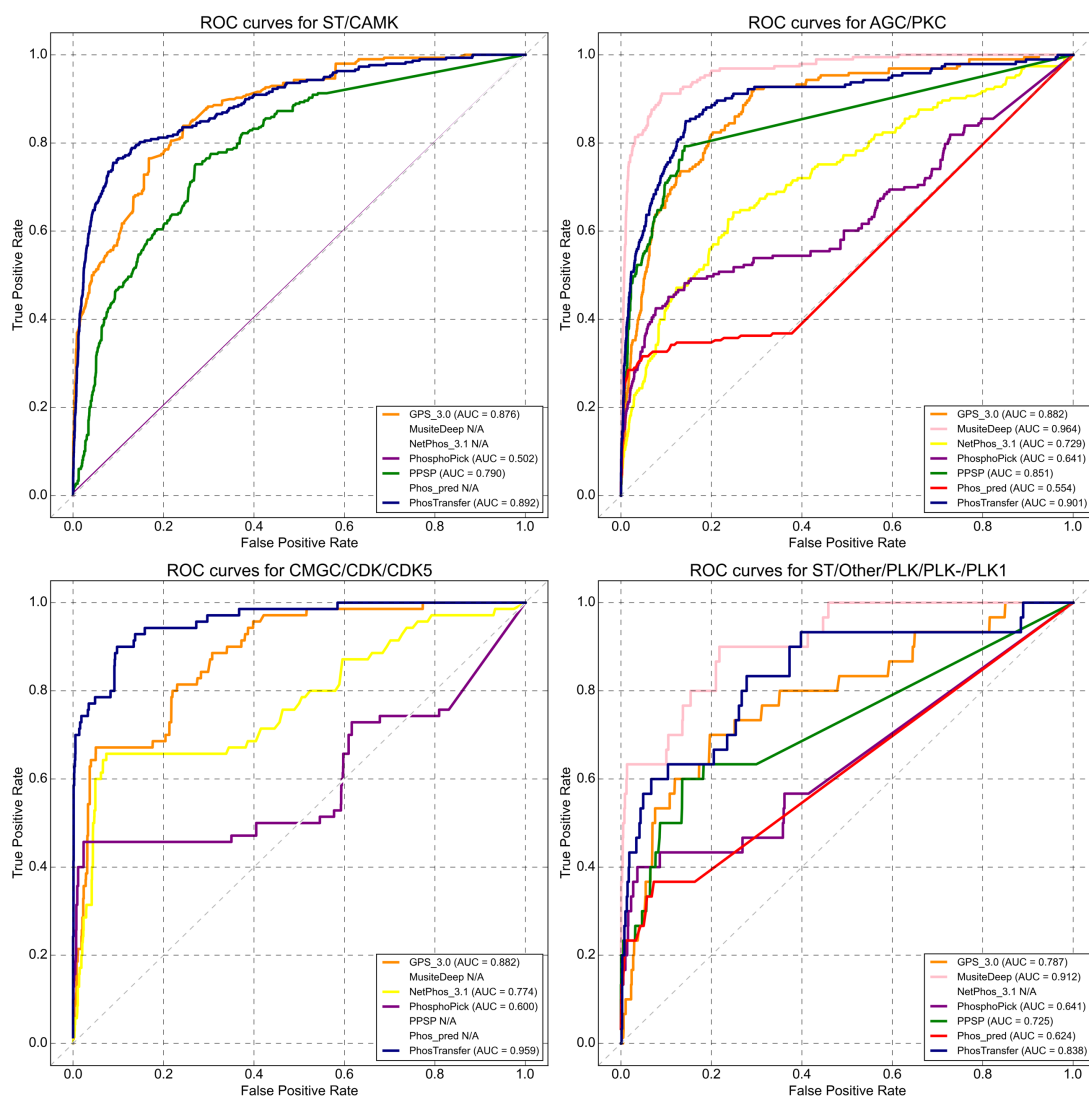
\* Model M and M' represent PhosTransfer and GPS 3.0 respectively.

\* Kinase nodes for which M and M' achieved equal performance were omitted from the counting.

Results in **Table 7.7** demonstrate that PhosTransfer achieved better performances for more kinase nodes in groups AGC and Other, for a comparable number of kinase nodes in groups CK1 and CMGC, and for fewer kinase nodes in groups Atypical, CAMK, STE and TK, as compared to GPS 3.0. Here, groups AGC and Other had 2631 and 1131 annotated phosphosites, respectively, while among the four groups in which PhosTransfer achieved inferior performance for more kinase nodes, group Atypical and STE had 224 and 241 annotated phosphosites, respectively. This result validated the intuition that a less satisfying performance by PhosTransfer can be expected if there are insufficient phosphosites for the kinase groups themselves. In theory, when deep transfer learning is applied, the first or second hidden layers are trained based on phosphosites of groups and reused later as the first or second feature extractors for models of families, subfamilies and PKs. Therefore, if the first or second convolutional feature extractor is trained with overfitting, the performance of models for families, subfamilies and PKs in this group will be negatively affected. This theoretically explains the relatively unsatisfying performance of PhosTransfer for groups Atypical and STE.

We further compared the prediction performance of PhosTransfer and GPS 3.0 to other phosphorylation site prediction methods including MusiteDeep [78], KinasePhos 2.0 [219], NetPhos 3.0 [35, 154], PhosphoPick [220], PPSP [354], and phos\_pred [355]. Feature groups and algorithms of compared methods are listed in **Table 2.2**. Considering the inconsistent sets of kinases that are available in the different prediction methods, we selected the group CAMK, family AGC/PKC, subfamily CGMC/CDK/CDK5, and PK Other/PLK-/PLK1 as the representative kinase nodes from each level of the HKCT. **Figure 7.16** demonstrates the ROC plots and corresponding AUC scores of different prediction methods for each of the kinase nodes. The results show that PhosTransfer achieved superior performance than most compared methods except for MusiteDeep, which, compared to PhosTransfer, also incorporated structural information and physicochemical properties as additional input features.





**Figure 7.16** Prediction performance comparison between PhosTransfer and several existing kinase-specific phosphorylation site prediction methods.

### 7.6.11 Validation of proposed methods for small datasets

In this thesis, we proposed three methods for kinase-specific phosphorylation site prediction, i.e. PhosContext2vec, DeepS2P-Ph and PhosTransfer. PhosContext2vec takes generated evolutionary information, predicted structural information and calculated physicochemical properties as input features and uses a generic shallow model, Support Vector Machine, to address the prediction task. In comparison, DeepS2P-Ph takes generated evolutionary information as input feature and uses a deep convolutional neural network to model the prediction task. In theory, this method involves more parameters than PhosContext2vec, which may result in decreased performance when only small training datasets are available. As a result, PhosTransfer was introduced in this section to address this issue by incorporating the hierarchical relationship between kinases and using transfer learning. It takes generated evolutionary information as input feature and uses the deep convolutional network with transfer learning to model the prediction task. Compared with the previous two methods, PhosTransfer not only takes advantage of deep architecture but also addresses the issue of overfitting. Therefore, it is suitable for small training datasets.

To validate the performance of these three methods on small training datasets, we conducted a group of comparison experiments on ST/Other/PLK/-/PLK1, which contains 108 and 30 phosphosites in the training and independent test datasets, respectively. Negative sampling was performance on the training dataset but not on the test dataset and the prediction performance was evaluated on the independent test dataset in terms of AUC scores. Note that the performance of each of these methods was selected based on hyper-parameter tuning. Please refer to **Section 7.4.6**, **Section 7.5.3** and **Section 7.6.5** for more details about hyper-parameters of these three methods, respectively. **Table 7.8** demonstrated the prediction performance for PhosContext2vec, DeepS2P-Ph and PhosTransfer on ST/Other/PLK/-/PLK1.

**Table 7.8** Prediction performance of PhosContext2vec, DeepS2P-Ph and PhosTransfer on ST/Other/PLK/-/PLK1 in terms of AUC score.

	AUC scores
PhosContext2vec	0.760
DeepS2P-Ph	0.632
PhosTransfer	<b>0.838</b>

The results in **Table 7.8** demonstrates that, due to the issue of overfitting, deep learning models do not always perform better than shallow models, especially on small training datasets. However, for certain tasks where there is additional domain

knowledge, e.g. the hierarchical relationship between protein kinases, transfer learning can be used to lower the requirement for excessive training data.

#### 7.6.12 Conclusion

In this section, we introduced the deep transfer learning framework DeepTransfer for kinase-specific phosphorylation site prediction. This framework was inspired by the hierarchical classification system of kinases and transfer learning. The basic idea is that phosphorylation sites of protein kinases within the same subfamily, family and group are likely to share similar local sequential and structural patterns. Therefore, models trained for higher level kinase nodes, which have more sufficient training data, can be transferred (reused) as feature extractors for lower level kinase nodes. When combined with deep learning, this idea is implemented in form of convolutional neural networks with multiple hidden layers, where each layer was trained individually based on the annotated phosphosites of kinase groups, families, subfamilies and PKs that are on the same tree path.

Results demonstrated that the prediction performance for most kinase nodes was improved when deep transfer learning was applied. In groups AGC, Atypical and Other, the positive correlation between the prediction performance and the number of pre-trained hidden layers was well demonstrated. When compared to the state-of-the-art phosphorylation site prediction methods, PhosTransfer achieved superior performance for various kinase groups, families, subfamilies and PKs, especially for kinase nodes in groups AGC and Other.

According to our investigation, the improved performance achieved by PhosTransfer is affected by the following factors. Firstly, PhosTransfer achieved more significant performance improvement for kinase nodes with limited quantities of training data. This is evidenced by the negative correlation between the performance improvement rate (PIR) and the number of annotated phosphosites in **Figure 7.14**. This result suggests that the application of PhosTransfer did help with the issue of overfitting during the

process of model training for kinase nodes with insufficient training data. Secondly, the number of phosphosites of the group itself can affect the prediction performance for its descendant families, subfamilies and PKs. The basic idea of PhosTransfer is that the models trained for kinase nodes (especially, kinase groups) with more sufficient training data can be reused as feature extractors for kinase nodes with insufficient training data. However, if the kinase group itself does not have enough annotated phosphosites, this idea would not work properly. Thirdly, it remains a challenge to predict phosphorylation sites for kinases whose annotated phosphorylation sites demonstrate diverse local patterns. According to [35], PROSITE motifs could recognize only 10% of annotated Y phosphorylation sites [68], which may explain the unsatisfactory performance of different prediction methods, including PhosTransfer, for kinases in group TK.

## 7.7 Chapter Summary

In this chapter, we demonstrated the positive effect of incorporating context2vec as a complementary feature vector in general and kinase-specific phosphorylation site prediction, the limitation of the DeepS2P framework in kinase-specific phosphorylation site prediction when there is a lack of training data, and the application of DeepTransfer to address the issue by using a level-by-level feature extraction strategy.

Firstly, we used context2vec as a complementary feature vector to existing residue-level biological representations of protein sequences in both general and kinase-specific phosphorylation site prediction. The distributed representation of sequence contexts at the residue-level represents a more general and contextual information than existing residue-level biological representations, such as homology-profiles, physicochemical properties, and structural information, that are generated for each individual residue without any consideration of local contexts. It also represents a more specific representation than sequence-level representations, such as ProtVec, that are generated for the whole sequence. The residue-level biological representations of individual residues, the residue-level distributed representation of local contexts, and the

representation of whole protein sequences or segments respectively represents the cases with contextual window size 0,  $m$  and infinite, where  $m$  defines the length of the local contexts based on which the context2vecs were generated. This application also validated the conclusion from the **Chapter 5** that the distributed representation of protein sequences should be used in combination with existing biological representations in order to achieve improved performance.

Secondly, we applied the DeepS2P framework for predicting kinase-specific phosphorylation sites. Despite the successful application of the DeepS2P framework to IDP/IDR and SSP prediction in **Chapter 6**, it achieved less favourable performance in phosphorylation site prediction, especially for kinases that have limited number of annotated phosphosites.

Finally, the less satisfying results achieved in **Section 7.5** inspired the application of the DeepTransfer framework to kinase-specific phosphorylation site prediction. We used the 4-level hierarchical classification system of protein kinases that was introduced in GPS 2.0 (which classifies protein kinases into groups, families and subfamilies), and trained a model for each of the tree node. Considering the basic assumption that protein kinases that belong to the same subfamilies, subfamilies belong to the same families and families belong to the same groups are more likely to catalyze phosphorylation sites with similar patterns. Therefore, reused the hidden layers of models of the parent node in the hierarchical classification system as the feature extractor for models of its children nodes. Results demonstrated that the feature vectors of protein kinases within different groups, families and subfamilies are separated better and better with the increase of the number of hidden layers (feature extractors) (see **Figure 7.15**).

The application of machine learning is always about the balance of between the involvement of domain knowledge and annotated data. In context2vec-based phosphorylation site prediction, the residue-level biological representations are generated with biological heuristics, which, to some extent, represents the involvement of domain knowledge. In contrast, the distributed representation of local sequence

contexts is generate purely based on large amount of protein sequence data without any biological heuristics. The successful combination of the two further validated the importance of the combination of domain knowledge and annotated data. In the application of the DeepS2P framework and the DeepTransfer framework, the former represents a framework design purely relying on annotated data. However, when there is a lack of annotated data, it failed to deliver satisfying prediction performance. The DeepTransfer framework was constructed base on the DeepS2P framework but involved domain knowledge such as the hierarchical classification system which is validated by domain expert. Therefore, in the application of deep learning techniques in biological data analysis, it crucial to incorporating biological background or domain knowledge to compensate the lack of annotated data.

# 8 Drug-target interaction prediction

## 8.1 Introduction

In **Section 4.1.4**, we introduced the application of distributed representation of protein sequences in three different scenarios. To validate the effectiveness of prot2vec in biological interaction level predictions, in this section, we systematically analyse the performance of prot2vec in the task of drug-target interaction (DTI) prediction.

Firstly, we provide an overview of DTI prediction in terms of the basic assumption, the most widely used data sources, and the methodology of performance evaluation, that are widely adopted by most machine learning based DTI prediction.

Secondly, we conduct comprehensive comparisons among the state-of-the-art machine learning models and the state-of-the-art protein/drug kernels for DTI prediction, from which we select the best performing model and the best performing drug kernels for further investigation of protein kernels.

Thirdly, we introduce in detail the methodology of prot2vec-based DTI prediction. We select the best performing prot2vec-based protein kernels for DTI prediction, by comparing the three implementations of prot2vecs and the six kernel functions that are applied to different implementations of prot2vecs.

Finally, we analyse the performance of prot2vec-based protein kernels and the state-of-the-art protein kernels in DTI prediction, demonstrating the superior performance of prot2vec as the kernel feature vector for DTI prediction.

## 8.2 An overview of drug-target interaction prediction

The assumption behind most machine learning-based methods for DTI prediction is that similar drugs tend to target similar proteins and vice versa [231, 234, 237, 238]. Therefore, the selection of the similarity scores for drugs (drug kernel) and proteins

(protein kernel) and the selection of machine learning models are the two key factors that affect the prediction performance of DTI prediction.

On one hand, each similarity score represents a potential factor that may be related to the process of drug docking, and the selection of the right similarity score is effectively the process of feature selection. The most widely used similarity scores are the chemical substructure similarity for drugs and the amino acid sequence similarity for proteins [233-240, 246]. Other similarities include 2D/3D fingerprints [246, 254], gene expression responses [246], side effects [246], and ATC annotations [246] similarities for drugs and protein-protein interaction (PPI) [246], GO annotation [246] and protein classification/hierarchy [254] similarities (See **Table 2.3**).

On the other hand, protein similarities and drug similarities are incorporated in various ways depending on the selection of the machine learning models. For kernel-based methods, similarities are represented as kernel matrices [254, 356]. For probabilistic graphical models, similarities are represented as nodes [234] or structure of the graphical model. While for neural networks, similarities are represented as vectors [253]. These similarities are used as input features to train the machine learning-based models, which are used later to predict novel DTI given the required similarities.

In **Section 2.3.4**, we introduced seven representative machine learning-based methods for DTI prediction. In this chapter, we perform comprehensive experiments to select the best performing drug kernel and the best performing machine learning method, based on which we further evaluated the prediction of prot2vec-based proteins in comparison with existing state-of-the-art protein kernels for DTI prediction.

## 8.3 Experimental design

### 8.3.1 Data sources and datasets

According to the previous works, data sources harbouring drug-target interactions include the KEGG BRITE [95] database, the DrugBank [97] database, the BRENDA [357]





sources including the KEGG BRITE database, the Drugbank database, the BRENDA database, and the SuperTarget database. The Yamanishi dataset is categorised into four subsets in terms of the protein family of the target proteins, including *enzymes (e)*, *ion channels (ic)*, *G protein-coupled receptors (gpcr)* and *nuclear receptors (nr)*. While the Perlman dataset was extracted from the KEGG BRITE database, the Drugbank database and the DCDB database. Compared to the Yamanishi dataset, the Perlman dataset incorporates more types of drug and protein similarities but does not distinguish among different target protein families. In order to perform independent tests, we constructed independent test benchmark KEGG-DTI by extracting drug-target interaction annotations from the KEGG BRITE database (downloaded in April 2015).

**Table 8.1** Statistical summary of proteins, drugs and DTIs in the KEGG, Yamanishi, and Perlman datasets

No. proteins/No. of drugs/No. of interactions		KEGG-DTI	Yamanishi	Perlman
Protein family	Enzymes ( <b>e</b> )	242/720/1809	664/445/2926	250/315/78
	Ion channels ( <b>ic</b> )	113/371/2825	204/210/1476	
	G protein-coupled receptors ( <b>gpcr</b> )	119/1290/3071	97/223/635	
	Nuclear receptors ( <b>nr</b> )	19/358/461	26/54/90	

\* Each cell in this table represents the number of proteins/the number of drugs/the number of interactions.

### 8.3.2 Cross-validation under four experimental settings

According to DTI prediction in real-world scenarios, there are four experimental settings for DTI prediction.

- 1) To predict DTIs between known drugs and known proteins;
- 2) To predict DTIs between known drugs and new proteins;
- 3) To predict DTIs between new drugs and known proteins; and
- 4) To predict DTIs between new drugs and new proteins.

Note that ‘known’ refers to the drugs or proteins that have at least one known DTI interaction, while ‘new’ refers to the drugs or proteins that have no known DTI predictions.

For methods using interaction profiles, such as BLM, SemEP and RBM, setting **4)** does not apply, since these methods rely on known DTIs to predict new DTIs of a protein (or a drug). For SemEP, setting **2)** and **3)** does not apply, since drugs or proteins without known interactions cannot be considered in the clustering process. For RBM, the predictive model is constructed for each protein, which requires that the queried protein's interaction information should be at least partially known. As a result, setting **2)** is not applicable. **Figure 8.2** illustrates the four settings used for our cross-validation, where black '1's represents the know interactions and the red '1's represent the predicted interactions.

	d1	d2	d3	d4	d5
p1	0	1	0	0	1
p2	0	1	0	0	0
p3	1	0	0	1	0
p4	1	0	1	0	0

(a) Interaction matrix for setting 1)

	d1	d2	d3	d4	d5	d1'
p1	0	1	0	0	1	1
p2	0	1	0	0	0	0
p3	1	0	0	1	0	1
p4	1	0	1	0	0	0

(b) Interaction matrix for setting 2)

	d1	d2	d3	d4	d5
p1	0	1	0	0	1
p2	0	1	0	0	0
p3	1	0	0	1	0
p4	1	0	1	0	0
p1'	0	0	1	1	0

(c) Interaction matrix for setting 3)

	d1	d2	d3	d4	d5	d1'	d2'
p1	0	1	0	0	1		
p2	0	1	0	0	0		
p3	1	0	0	1	0		
p4	1	0	1	0	0		
p1'						0	1
p2'						0	0

(d) Interaction matrix for setting 4)

**Figure 8.2** Drug-target interaction prediction under four experimental settings

We performed five-fold cross-validation (5-fold-CV) and leave-one-out cross-validation (LOOCV) for the four settings, respectively. For 5-fold-CV, we divide the dataset into 5 folders by protein (*cv-by-proteins*), by drug (*cv-by-drugs*) and by

interactions entries (*cv-by-interactions*), which correspond to the setting **2**), **3**) and **1**) respectively. For LOOCV only one interaction entry was left out each time, leaving partial interaction for both drug and protein, which makes LOOCV an experimental solution for setting **1**).

### 8.3.3 Performance evaluation

To compare the prediction performance of different machine learning models, we chose the area under the ROC curve (AUC) and the area under the precision-recall curve (AUPR) as the major evaluation measures. AUPR can better interpret the prediction performance of different methods on unbalanced datasets [360], while AUC is one of the most widely used measures for evaluating DTI prediction tasks [231, 253]. In addition, to measure the time complexity of these models, we also recorded the training time over different folds from which the average training time was calculated. In terms of independent tests, the predicted novel drug-target interactions were manually validated against the latest KEGG database, the ChEMBL database and the DrugBank database [233, 235, 237].

## 8.4 DTI prediction based on existing machine learning methods

### 8.4.1 Comparison among machine learning based methods

In **Section 2.3.4**, we introduced seven representative machine learning-based methods for DTI prediction. They are PWK [254], BLM [235], KBMF2K [233], GIP+WNN, PSL, SemEP [255] and RBM [253], representing kernel-based methods [240] [236] [254] [236], the local model [236], the K-nearest neighbour method [237], Bayesian networks [233], statistical relational models [234], the clustering method [255] and neural networks [253], respectively. In this section, the prediction performance of the seven machine learning - based methods was evaluated on the subset **nr**, **ic**, **e**, and **gpcr** in the *Yamanishi* dataset

using 5-fold cross-validation. **Table 8.2** illustrates the average AUC scores with a significant level of 95% for different machine learning-based methods was reported.

**Table 8.2** Performance comparison among different machine learning methods for drug-target interaction prediction on the Yamanish dataset.

		Method	nr (26x54)	ic (204x210)	e (664x445)	gpcr (97x223)
AUC	cv-by-interactions	PWK	0.555±0.047	0.567±0.007	N/A	0.594±0.014
		BLM	0.510±0.000	0.554±0.000	0.519±0.000	0.548±0.000
		KBMF2K	<b>0.921±0.007</b>	<b>0.978±0.003</b>	<b>0.983±0.002</b>	<b>0.953±0.006</b>
		GIP-WNN	0.908±0.020	0.978±0.004	0.974±0.003	0.950±0.005
		PSL	0.566±0.025	0.791±0.002	0.759±0.007	0.768±0.013
		SemEP	0.780±0.029	0.855±0.002	0.826±0.005	0.816±0.004
		RBM	0.541±0.023	0.931±0.005	0.917±0.003	0.814±0.012
	cv-by-proteins	PWK	0.458±0.054	0.606±0.013	N/A	0.512±0.025
		BLM	0.521±0.078	0.927±0.010	0.929±0.009	0.842±0.030
		KBMF2K	<b>0.731±0.022</b>	<b>0.944±0.007</b>	<b>0.943±0.003</b>	0.848±0.043
		GIP-WNN	0.656±0.090	0.936±0.004	0.934±0.012	<b>0.872±0.036</b>
		PSL	0.596±0.001	0.817±0.010	0.744±0.013	0.605±0.015
		SemEP	0.510±0.000	0.554±0.000	0.519±0.000	0.548±0.000
		RBM	0.354±0.019	0.684±0.007	0.517±0.008	0.495±0.015
	cv-by-drugs	PWK	0.504±0.015	0.565±0.021	N/A	0.575±0.011
		BLM	0.801±0.036	0.720±0.030	0.778±0.008	0.768±0.024
		KBMF2K	0.817±0.023	<b>0.794±0.006</b>	0.819±0.026	0.839±0.018
		GIP-WNN	<b>0.826±0.071</b>	0.764±0.028	<b>0.850±0.034</b>	<b>0.856±0.016</b>
		PSL	0.721±0.017	0.729±0.011	0.688±0.021	0.781±0.013
		SemEP	0.510±0.000	0.554±0.000	0.519±0.000	0.548±0.000
		RBM	0.497±0.017	0.517±0.010	0.524±0.021	0.561±0.013
Time (s)	cv-by-interactions	PWK	1.281	600.058	N/A	127.46
		BLM	7.888	61.905	203.044	28.966
		KBMF2K	13.853	164.590	1600.600	103.760
		GIP-WNN	<b>0.183</b>	<b>1.547</b>	<b>20.597</b>	<b>1.175</b>
		PSL	2.351	30.355	291.949	13.628
		SemEP	0.242	6.203	26.040	1.990
		RBM	2.676	25.215	183.328	13.611

\* The best performance in terms of AUC scores and time consumption for each dataset was highlighted as bold.

According to the results in **Table 8.2**, KBMF2K and GIP-WNN achieved the highest AUC values with no significant differences between each other. However, KBMF2K had smaller standard errors over all four datasets and all three settings, suggesting that KBMF2K performs more robustly, in regardless of the data size and data distribution. Theoretically, the performance of GIP-WNN relies on the distribution similarity between training data and test data, which may explain the relatively larger standard errors. The BLM method performed worse than KBMF2K and GIP-WNN, significantly. It is equivalent to KBMF2K with subspace dimension  $R = 2$ , while in KBMF2K,  $R=25$  has been chosen as the optimised parameter value from the model selection [233], which gives KBMF2K the winning edge. On the other hand, BLM only relies on drug or protein's known interactions for prediction, while GIP-WNN also considers similarities among drugs and similarities among proteins for prediction. These differences may explain the better prediction performance achieved by GIP-WNN. However, as a nearest-neighbour method, GIP-WNN loses to BLM in terms of robustness as suggested by BLM's smaller standard error.

According to the reported results in **Table 8.2**, we can obtain the following observations. Firstly, methods with higher AUC scores require more time for training, except for GIP-KNN which achieved higher performance within a shorter period of processing time. Secondly, the prediction performance also depends on the size of the dataset used for training the model. For example, most of the compared methods achieved less satisfying prediction performance on the **nr** dataset, which has the least number of training examples. On datasets with more sufficient training data, such as **ic**, **e** and **gpcr** datasets, the prediction performance of most of the models were improved. Thirdly, it is more likely to achieve better prediction performance under setting cv-by-interactions than under the other two experimental settings. In setting cv-by-proteins and cv-by-drugs, certain numbers of proteins and drugs do not have any interaction information. In contrast, all proteins and drugs have at least one known drug-target

interaction under the setting cv-by-interactions. This difference may explain the better prediction performance achieved under setting cv-by-interactions.

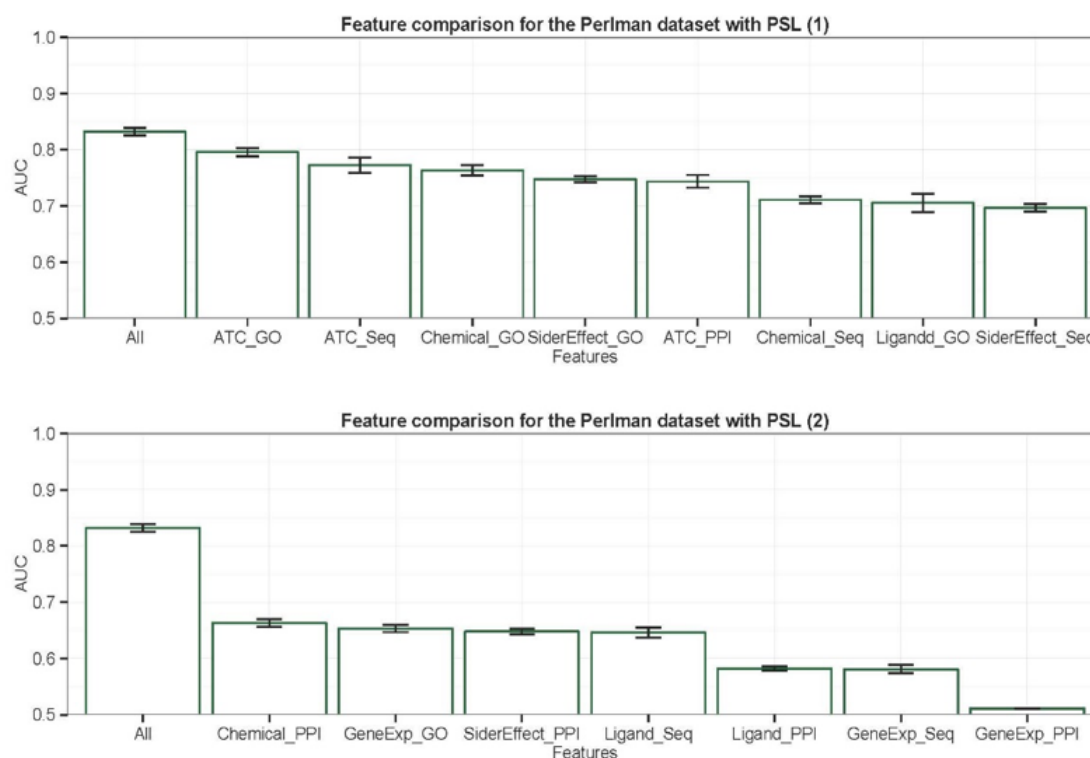
#### 8.4.2 Comparison among existing drug/protein kernels

Kernel refers to the similarity function applied on pairs of objects, such as data points, examples, individuals, etc. Therefore, the similarities among two drugs and the similarity among two proteins are also referred to as drug kernel and protein kernels, respectively. In this section, different protein kernels and drug kernels were experimentally compared to each other, from which the best protein kernel and the best drug kernel were selected and used in further investigations.

The comparison among different drug/protein kernels was conducted on the Perlman dataset. The Perlman dataset contains 5 similarity features for drugs, which are Chemical, Ligand, Gene Expression, Side Effect and ATC, and 3 similarity features for proteins, which are Sequence, PPI and GO (see **Section 2.3.4**). For most of the existing machine learning methods for DTI prediction, it only requires one pair of similarity features from the drug and protein spaces respectively. Therefore, we combine each protein kernel with all drug kernels to form 15 drug/protein kernel pairs. Note that PSL is the only method that can combine multiple pairs of features. In order to demonstrate the difference between different drug/protein kernel pairs and the difference between using single drug/protein kernel pair and using multiple drug/protein kernel pairs, we only show the results achieved by PSL.

We collected the AUC scores and their 95% significant standard errors of PSL under 5-fold cross-validation by interactions, then order them in descending order, which leads to the results demonstrated in **Figure 8.3**.





**Figure 8.3** Comparison among different DTI drug/protein kernel pairs.

According to the results shown in **Figure 8.3**, the best prediction performance was achieved using the annotation-based drug kernel ATC and the annotation-based protein kernel GO, with an AUC score of 0.798. Among all drug kernels, the ATC kernel represents one of the best performing kernels as evidenced by its strong performance when combined with all three protein kernels GO, SEQ and PPI. The Chemical kernel and the SiderEffect kernel also achieved satisfactory prediction performance when combined with two protein kernels GO and SEQ. Among all protein kernels, the GO outperformed the other two, appearing four times in the eight highly ranked drug/protein kernel pairs. In comparison, the SEQ kernel appeared three times while the PPI kernel only appeared once.

Overall speaking, three drug kernels and two protein kernels appeared at least twice in the eight highly ranked drug/protein kernel pairs. They are ATC, Chemical and SiderEffect for drugs, and GO and Sequence for proteins. Note that only one of them are

*protein inter-domain* -based features, *i.e.* the drug kernel SiderEffect. In contrast, *annotation*-based features and *sequence* or *structure*-based kernels achieved superior performance for DTI prediction. Furthermore, a significantly better AUC score of 0.832 was achieved by combining multiple pairs of drug/protein features, which is denoted as ALL in **Figure 8.3**.

#### **8.4.3 New interaction prediction with existing methods and kernels**

To further evaluate the robustness and scalability of the reviewed methods, in this section we conducted an independent test under the four settings using a carefully curated test dataset without overlap with the datasets for training the methods.

Previous studies only conduct the independent test under setting **1**), where new DTI new drug-target interactions are predicted from a given interaction network with LOOCV, *i.e.* within known proteins and known drugs contained in the Yamanishi dataset. In this section, we construct a new test dataset KEGG-DTI from the latest version of KEGG (downloaded on 2015-12-16). Since new interactions were predicted based on similar drugs' or proteins' interaction with existing methods, known DTIs have to be given as evidence when the prediction is performed. Therefore, we built our new dataset based on the Yamanishi dataset, where new proteins and new drugs (that do not exist in the Yamanishi dataset) are added from the latest KEGG.

Given only the DTIs that were contained in the *Yamanishi* dataset, we predicted new interactions between known proteins and known drugs, new proteins and known drugs, new drugs and known proteins and new proteins and new drugs, which correspond to the setting **1) 2) 3) and 4)** respectively. Finally, the top 30 predicted novel interactions were compared to the latest version of the KEGG database to manually evaluate the correctness of the novel predictions.

**Table 8.3** Performance of different machine learning methods for DTI prediction under four experimental settings, evaluated using TP/P and precision (PR).

	Setting 1)		Setting 2)		Setting 3)		Setting 4)	
	TP/30	PR	TP/30	PR	TP/30	PR	TP/30	PR
BLM	8/30	0.267	12/30	0.400	<b>30/30</b>	<b>1.000</b>	n/a	n/a
KBMF2K	7/30	0.233	6/13	0.412	26/30	0.867	1/1	<b>1.000</b>
GIP-WNN	<b>9/30</b>	<b>0.300</b>	12/30	0.400	28/30	0.933	9/30	0.300
PSL	2/30	0.067	<b>14/30</b>	<b>0.467</b>	28/30	0.933	<b>25/30</b>	0.833
SemEP	5/30	0.167	n/a	n/a	n/a	n/a	n/a	n/a
RBM	5/30	0.167	0/30	0.000	0/30	0.000	2/30	0.067

\* The best performance in terms of precision was highlighted as bold.

According to the results in **Table 8.3**, GIP-WNN, PSL and BLM achieved the highest accuracy under setting 1), 2) and 3), respectively. Under setting 4), where interactions were predicted between new proteins and new drugs, KBMF2K achieved the best precision of 1.0 but with only one prediction, while PSL correctly predicted 25 out of 30 DTIs. One feature of KBMF2K is that it tends to give fewer predictions but with higher precision, as is shown in setting 2) and 3), where only 13 and 1 predictions are made in total, while other methods make more predictions but with more false positives. It explains KBMF2K's overall better AUC results in the previous cross-validation. PSL performed surprisingly well compared to its performance in cross-validation. It achieved 25 correct predictions and a precision of 0.833 under setting 4), which shows its ability to generalize to new proteins and new drugs.

In this independent test, we adopted the *open world assumption* [361], in which the interactions that are not annotated as positive are considered to be unknown instead of negative. Accordingly, we did not calculate the sensitivity score because, in the real world, no data source guarantees complete drug-target interaction.

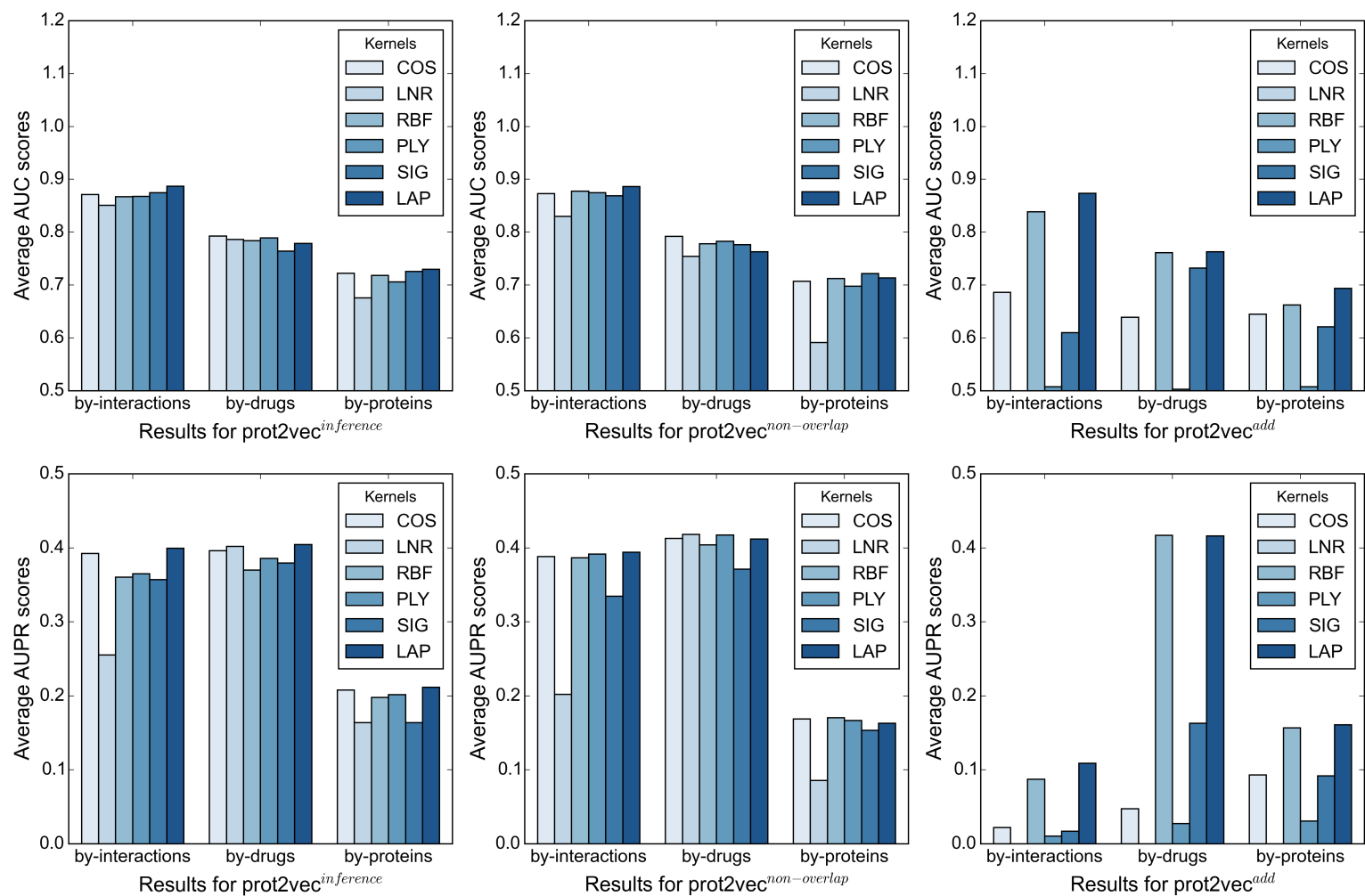
## 8.5 DTI prediction with prot2vec-based protein kernels

To validate the effectiveness of the prot2vec as the kernel feature vector for biological interaction-level prediction tasks, we applied different kernel functions to generate the similarity between pairs of proteins, resulting in different prot2vec-based protein kernels.

In order to demonstrate fairness among various protein kernels that are evaluated and compared in the following experiments, another two key components for DTI prediction was specified according to the overall evaluation performed in **Section 8.4.1** and **8.4.2**. We used the structure-based drug similarity *Chemical* as the drug kernel and the *KBMF2K* method as the predictive model for DTI prediction. With the drug kernel and predictive model specified, each of the evaluated protein kernels is combined with the selected drug kernel to form a drug/protein kernel pair, which is later used as the input feature for the selected predictive model.

#### 8.5.1 Comparison among different kernels generated from prot2vec

In this comparison, the prediction performance of different kernel functions, including *Linear*, *RBF*, *Polynomial*, *Cosine*, *Laplacian*, and *Sigmoid*, was evaluated and compared. Each of the six kernel functions were applied to three different implementations of prot2vecs, including prot2vec<sup>inference</sup>, prot2vec<sup>non-overlap</sup> and prot2vec<sup>add</sup>. Therefore, 18 protein kernels were generated and evaluated for DTI prediction. For all the six compared kernels, all the three implementations of prot2vecs, and all the three experimental settings, 5-fold cross-validation was performed on the Perlman dataset (see **Section 8.3.1**), respectively. **Figure 8.4** demonstrates the average AUC and AUPR scores of the 18 different prot2vec-based protein kernels for DTI prediction.



**Figure 8.4** Performance comparison among different kernels that were generated from prot2vecs.

According to the AUC scores demonstrated in **Figure 8.4**, the Laplacian kernel performed the best for all three implementations of prot2vecs under the experimental setting **1)**; the Cosine kernel performed the best for prot2vec<sup>inference</sup> and prot2vec<sup>non-overlap</sup> while the Laplacian and RBF kernel performed the best for prot2vec<sup>add</sup> under setting **3)**; and the Laplacian kernel performed the best for prot2vec<sup>inference</sup> and prot2vec<sup>add</sup>, and the Sigmoid kernel performed the best for prot2vec<sup>non-overlap</sup> under setting **2)**. Among all 6 compared kernel functions, the linear kernels generated from all three implementations of prot2vecs performed the worst in all three experimental settings. The AUPR scores of different kernels demonstrated similar patterns in terms of the performance difference among different kernels.

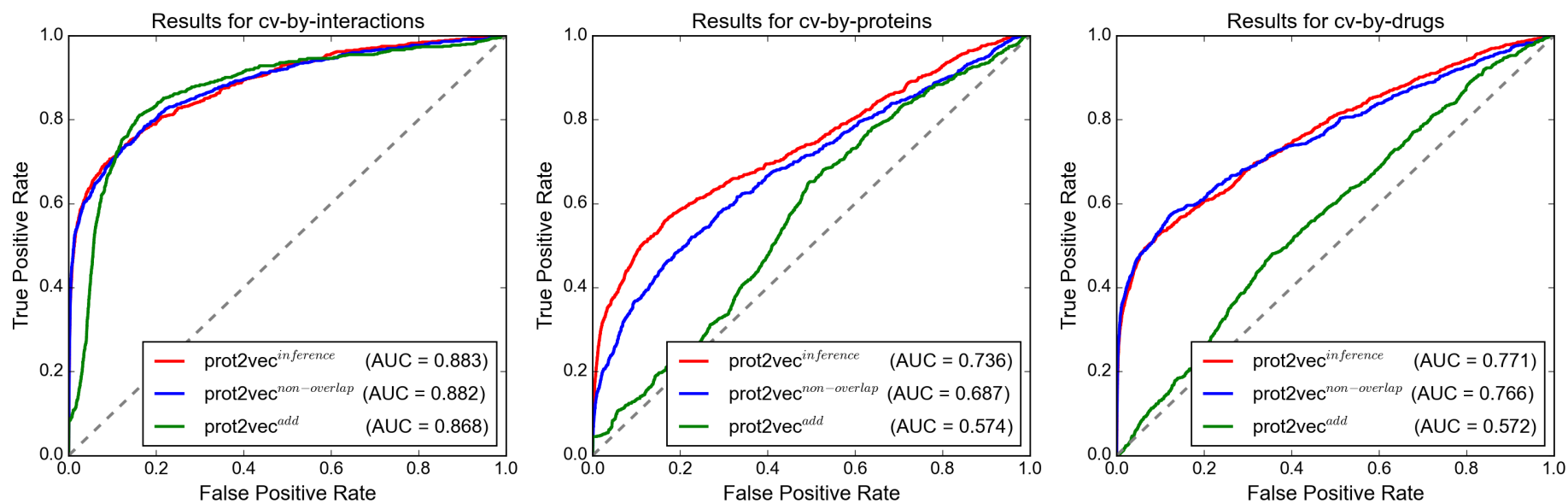
In general, kernels generated from prot2vec<sup>inference</sup> and prot2vec<sup>non-overlap</sup> demonstrated similar prediction performance, for which the Cosine kernel and the Laplacian kernel outperformed other kernels in all three experimental settings. Also, the prediction performance of the various protein kernels generated from these two implementations is more consistent compared to that of the protein kernels generated from prot2vec<sup>add</sup>. For example, the prediction performance of the *Cosine*, *Linear*, *RBF*, *Polynomial*, *Sigmoid* and *Laplacian* kernels generated from prot2vec<sup>inference</sup> was 0.871, 0.850, 0.867, 0.868, 0.875 and 0.887, respectively, with a standard deviation of 0.012 under the setting **1)**. In contrast, the prediction performance of the above six kernels generated from prot2vec<sup>add</sup> was 0.686, N/A, 0.839, 0.507, 0.610, and 0.874, respectively, with a standard deviation of 0.154 under the same experimental setting.

In terms of the three experimental settings, the setting **1)** where the cross-validation was conducted over all drug-target interactions had better performance than setting **2)** and **3)**, under which the cross-validation was conducted over proteins and drugs, respectively. The only exception is for the AUPR scores calculated for protein kernels generated from prot2vec<sup>add</sup>, where the prediction performance under the setting **2)** is superior to that achieved under setting **1)** and **3)**.

The following observations can be obtained from the results in **Figure 8.4**. Firstly, better prediction performance was achieved by different protein kernels for different implementations of prot2vecs under different experimental settings. However, the Laplacian kernel and the Cosine kernel achieved superior performance than other kernels that were evaluated and compared. Secondly, the prediction performance of the 6 tested kernels generated from the prot2vec<sup>add</sup> demonstrated significant differences from each other than those generated from the prot2vec<sup>inference</sup> and prot2vec<sup>non-overlap</sup>. Thirdly, the prediction performance under setting **1)** was superior to those under setting **2)** and **3)**; and the prediction performance under setting **2)** was superior to the setting **3)**. This result is consistent with the observation that the number of interactions is larger than the number of drugs and the number of drugs is larger than the number of proteins. It indicates that the number of training examples is positively related to the prediction performance, over the three experimental settings.

### 8.5.2 Comparison among three implementations of prot2vecs

In **Section 4.1.2**, we introduced four implementations of the distributed representation of protein sequences. To evaluate the prediction performance of the different implementations of prot2vecs in DTI prediction, in this section, we conduct cross-validation tests to compare the prediction performance of the prot2vec<sup>add</sup>-based protein kernel, the prot2vec<sup>non-overlap</sup>-based protein kernel and the prot2vec<sup>inference</sup>-based protein kernel. **Figure 8.5** illustrates the ROC curves and corresponding AUC scores for the above three protein kernels under three experimental settings.



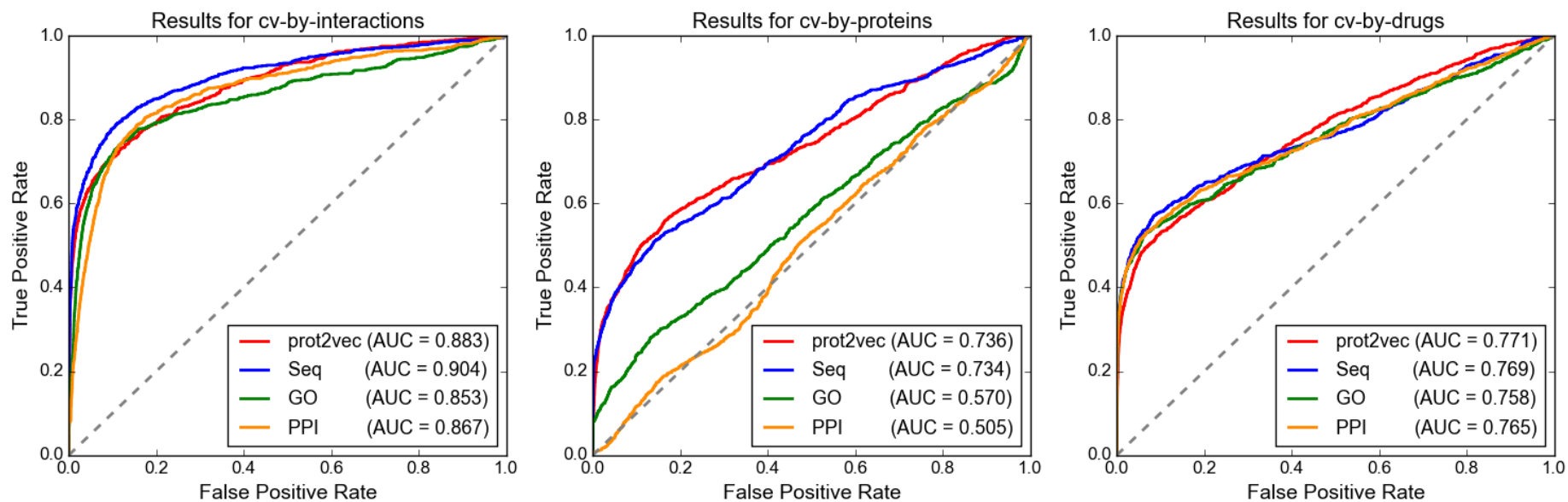
**Figure 8.5** Performance comparison among the three implementations of prot2vecs as the kernel feature vectors for DTI prediction.



In all three experimental settings, protein kernels generated from  $\text{prot2vec}^{\text{inference}}$  and  $\text{prot2vec}^{\text{non-overlap}}$  consistently outperformed those generated from  $\text{prot2vec}^{\text{add}}$ . Since the experimental setting cv-by-proteins relies solely on the selection of protein kernels, we specifically discuss the results for cv-by-proteins. According to the results shown in **Figure 8.5**, the best prediction performance was achieved by  $\text{prot2vec}^{\text{inference}}$ -based kernels, with an AUC score of 0.736, which improved the AUC score by 0.049 and 0.062 when compared with those achieved by  $\text{prot2vec}^{\text{non-overlap}}$ -based kernel and the  $\text{prot2vec}^{\text{add}}$ -based kernel. It suggests that the order of amino acids in protein sequences plays an important role in generating vector representations for DTI prediction. At the same time, the superior performance of the  $\text{prot2vec}^{\text{inference}}$ -based kernel over the  $\text{prot2vec}^{\text{non-overlap}}$ -based kernel showed that the overlapping splitting is better than the non-overlapping splitting for generating  $\text{prot2vecs}$ .

### 8.5.3 Comparison with the state-of-the-art protein kernels

In **Section 8.4.2**, three state-of-the-art protein kernels were evaluated and compared for DTI prediction. They are the sequence-based kernel *SEQ*, the inter-domain -based kernel *PPI*, and the annotation-based kernel *GO*. In this section, we validate the effectiveness of the  $\text{prot2vec}$ -based protein kernels by comparing their performance to those of the three state-of-the-art protein kernels in DTI prediction. Here, the protein kernel generated from the  $\text{prot2vec}^{\text{inference}}$  representations using the Laplacian kernel function was selected, since it achieved the best performance according to **Section 8.5.1**. **Figure 8.6** demonstrates the ROC curves with corresponding AUC scores for the compared protein kernels.



**Figure 8.6** Performance comparison among state-of-the-art protein kernels for drug-target interaction prediction.

According to the results in **Figure 8.6**, the prot2vec-based protein kernel and sequence-based kernel *SEQ* achieved superior performance than annotation-based kernel *GO* and inter-domain -based kernel *PPI*. For cv-by-proteins and cv-by-drugs experimental settings, prot2vec-based kernel improved the AUC score by 0.002 and 0.002, respectively, when compared with the state-of-the-art protein kernel *SEQ*. It achieved the second-best performance in setting cv-by-interactions. However, it is noteworthy that the prediction performance in setting cv-by-interactions was also affected by the selection of drug kernels. In contrast, the prediction performance in setting cv-by-proteins relies solely on the quality of protein kernels. Therefore, the performance in cv-by-proteins is of the most interest for investigating the performance of different protein kernels.

## 8.6 Chapter summary

In this chapter, we evaluated the effectiveness of prot2vec-based protein kernels for DTI prediction. We first conducted comprehensive benchmarking experiments to evaluate the prediction performance of seven existing machine learning-based methods, three protein kernels and five drug kernels. We found that among all machine learning-based methods, the KBMF2K method achieved the best prediction performance and the GIP-WNN method achieved comparable performance using the least execution time. While for protein and drug kernels, the better prediction performance was achieved by using sequence-based protein kernel *SEQ*, annotation-based protein kernel *GO*, structure-based drug kernel *Chemical*, annotation-based drug kernel *ATC*, and inter-domain -based drug kernel *SiderEffect*. According to these results, we selected the best performing machine learning-based method and the best performing drug kernel, with which we further evaluated the prediction performance of prot2vec-based protein kernels.

The distributed representation of protein sequences provides a kernel-function friendly representation for generating protein sequence-based kernels. With protein

sequences represented using three different implementations,  $\text{prot2vec}^{\text{inference}}$ ,  $\text{prot2vec}^{\text{non-overlap}}$  and  $\text{prot2vec}^{\text{add}}$  (e.g. ProtVec [193]), we respectively applied six different kernel functions to generate 21 protein sequence -based kernels and evaluated their prediction performance for DTI prediction. With the best performing kernel function that was selected from the six compared kernels functions, we further evaluated the prediction of  $\text{prot2vec}$ -based protein kernels in comparison with the existing state-of-the-art protein kernels that have been applied to DTI prediction.

According to the performance comparison for DTI prediction in **Figure 8.4**, **Figure 8.5**, and **Figure 8.6**, we obtained the following conclusions. Firstly, the order of amino acids in protein sequences plays an important role in the generation of  $\text{prot2vec}$ -based protein kernels. This is evidenced by the superior performance achieved by our  $\text{prot2vec}^{\text{inference}}$  and  $\text{prot2vec}^{\text{non-overlap}}$ -based protein kernels when compared with that achieved by  $\text{prot2vec}^{\text{add}}$ -based protein kernels. Secondly, the representation  $\text{prot2vec}$  provides a flexible solution for generating protein kernels using multiple kernel functions, from which a best performing kernel can be selected for real-world applications. In this chapter, we found that, among the six compared kernel functions, the LAP kernel and COS kernel performed better for DTI prediction when applied to  $\text{prot2vec}$  representations. Thirdly, compared to existing state-of-the-art protein kernels,  $\text{prot2vec}$ -based protein kernels achieved superior performance in DTI prediction. According to the evaluation results, when protein kernel plays a key role in the prediction,  $\text{prot2vec}$ -based kernels improved the AUC score by 0.166 and 0.231, respectively, compared to that of the annotation-based protein kernel GO and inter-domain -based protein kernel PPI. It also outperformed the state-of-the-art protein kernel SEQ for DTI prediction.

## 9 Conclusions and future work

Deep learning, as an emerging technique, has achieved breakthrough performance improvements in various application areas including image processing, natural language processing and speech recognition. In this thesis, we applied a variety of deep learning techniques to the analysis of protein sequences, structures and functions, focusing on the specialised issues and challenges that are specific to biological data analysis. We contributed to both the deep learning field by extending its application in designing domain-specific predictive frameworks and to the computational biology field by demonstrating the strong performance of deep learning techniques for biological data analysis. We also developed publically available bioinformatics tools for future research.

In previous chapters, we proposed, evaluated and analysed deep learning frameworks for predicting protein structures and functions directly from protein sequences. In this chapter, we conclude the thesis with a summary of heuristics for designing deep learning frameworks. Based on these conclusions, we discuss possible directions for future work.

### 9.1 Heuristics for designing deep learning frameworks

In this thesis, we explored the systematic design of deep learning framework for groups of prediction tasks and critically analysed the prediction performance of deep learning techniques in protein analysis. According to the designing process of the proposed deep learning frameworks and the evaluation process of their prediction performance, we developed a set of heuristics for designing deep learning frameworks in protein analysis.

### 9.1.1 Heuristic 1: Purely data-driven representation should be used in combination with biological heuristic-based representations

In natural language processing (NLP), the generation of distributed representations is unsupervised, purely data-driven and conveys high semantic relevance. It greatly simplifies the NLP pipelines by replacing the incorporation of NLP background knowledge with automatic unsupervised learning processes. As one of the more recently proposed distributed representations in NLP, *word2vec* [289] and *doc2vec* [169] have achieved superior performance compared with the traditional word embedding techniques and the generic heuristic-based NLP pipelines [277]. In this thesis, we demonstrated the performance of the purely data-driven representation, *prot2vec*, in protein analysis. We introduced three novel implementations of *prot2vec* and proposed their application to three different granularities of prediction tasks.

In **Chapter 5**, we introduced the application of *prot2vec* to protein family prediction. We found that the *prot2vec*-based prediction method *PfamProt2vec* achieved superior performance than the method employing the existing implementation of the distributed representations of protein sequences, *i.e.* *ProtVec* [193]. However, when compared with the HMMER 3.1 method which employs the homology profile-based representation, *PfamProt2vec* achieved superior precision scores but performed less favourably in terms of sensitivities and balanced accuracies.

We conducted correlation studies to investigate eight potential factors that may affect the prediction performance of *PfamProt2vec*. According to the results, we found that the prediction performance of *PfamProt2vec* was negatively affected by three factors, *e.g.* low sequential similarities within each protein family; low homological similarities within each protein family; and the large number of related families of each protein family. In contrast, the prediction performance of HMMER 3.1 was negatively affected only by the second factor among the three. This result indicates that the sequence-based *PfamProt2vec* is less accurate for protein families with less sequential similarities, while

the homology-based HMMER 3.1 is less accurate for protein families with less homology similarities. Therefore, the sequence-based distributed representation, prot2vec, should be used in combination with existing biological heuristic-based representations such as homology profiles, physicochemical properties and structural information.

Based on this conclusion, in **Section 7.4**, we predicted phosphorylation sites by combining the purely data-driven representation, context2vec, with biological heuristic-based representations. Results suggest that models using the combination of the two as the input feature achieved improved performance compared to those using either context2vec or the biological heuristic-based representations independently as the input feature. Specifically, the prediction performance in terms of AUC was significantly improved by 0.051, 0.033, 0.025, 0.108, 0.090 for site S, site T, kinase AGC/PKA, kinase CMGC/CDK, and kinase Other/CK2, respectively, when context2vec was used in addition to biological heuristic-based representations.

The prediction performance for protein family prediction and phosphorylation site prediction demonstrated the importance of combining purely data-driven representations and biological heuristic-based representations.

### **9.1.2 Heuristic 2: Biological background knowledge should be incorporated in the designing of deep learning frameworks for protein and biological predictive analysis**

The purpose of developing automated predictive models is to make novel and testable predictions with any resource that is available. Deep learning techniques are especially suitable for application areas with sufficient training data, based on which the structure and parameters of the predictive model are automatically learned during the process of model training. However, when there is a lack of data, domain knowledge represents another source of information that can be used to partially determine the structure of the predictive models. Typical machine learning approaches that allows for the incorporation

of domain knowledge include probabilistic soft logic [362], dynamic Bayesian networks [363] and Markov logic networks [127].

Protein predictive modelling suffers from a lack of structural/functional annotations, which causes the issue of a lack of training data for many protein sequence-based prediction tasks. However, it can benefit from biological background knowledge that is manually curated by biologists (e.g. the Gene Ontology [249]) and a large amount of experimental observations drawn from experimental analysis (e.g. the qualitative correlation between protein intrinsic disorder and secondary structure populations [66]). With these heuristics in mind, in **Section 4.3** and **Section 4.4**, we proposed the multitask framework, Multi-task and Cross-stitch, by incorporating correlations among protein structural properties, and the deep transfer learning framework, DeepTransfer, by incorporating the hierarchical classification systems of enzymes.

The rationale for designing the multitask frameworks is that protein structural properties are correlated, and the model learned for one structural property may provide evidential information for the prediction of the other structural property. This was validated by the results generated from the application of the Multi-task and Cross-stitch framework to the simultaneous prediction of protein IDP/IDRs and SSPs, in **Section 6.4**. The results of cross-validation tests indicated that the multitask predictive model Multi-task-D and Cross-stitch-D outperformed the singletask predictive model DeepS2P-D in terms of all four evaluation measurements including the cross entropy, the area under the ROC curve, Matthew's coefficients of correlation, and the balanced accuracy. The results of the case study for CASP9 target T520 demonstrated how the predictions of SSPs have positively affected the sensitivity in IDP/IDR prediction. Further, the results of the case studies performed for protein p53 and SOSSB1 provided the real-world application scenarios of multitask predictive models, from which biological interpretations of the prediction results can be made. In addition, the Cross-stitch-D model generated a quantitative correlation heat map between IDP/IDR prediction and SSP prediction, which



demonstrated the capability of the Cross-stitch framework in transforming qualitative correlations to quantitative correlations between protein structural properties.

The rationale for designing the DeepTransfer framework is that there exists hierarchical classification systems (trees) of enzymes and the models learned for tree nodes at a higher level can be transferred as the feature extractors for models of their descendant tree nodes. In **Section 7.6**, we specifically applied this framework to kinase-specific phosphorylation site prediction where there is a lack of training data for individual protein kinases. According to the results generated from correlation studies, we found that the prediction performance was more significantly improved for kinases with less training data, which indicated the effectiveness of the DeepTransfer framework in leveraging limited training data for rigorous model training.

Combining the improved performance achieved in the simultaneous prediction of IDP/IDRs and SSPs and the prediction of phosphorylation sites for kinases with limited training examples, we conclude that it is crucial to incorporate biological background knowledge in the design of deep learning frameworks for protein analysis, especially when there is a lack of training data.

### **9.1.3 Heuristic 3: Deep learning frameworks for protein analysis should be designed at a conceptual level**

As demonstrated in **Chapter 1** and **Chapter 2**, existing applications of deep learning to protein analysis were performed in an *ad hoc* manner. However, according to the research conducted in this thesis, the prediction tasks in protein analysis can be generalised to a more abstract level, based on which deep learning frameworks can be designed for groups of prediction tasks. This idea greatly improved the efficiency in constructing predictive models for different tasks.

In **Section 4.2**, we proposed the DeepS2P framework for predicting residue-level structural/functional properties based on protein sequences. It allows flexibility in the selection of input features, the determination of the final model structure (via hyper-

parameter tuning), and the format of the prediction outputs (such as the number of output values, numerical or categorical outputs, etc.), depending on the prediction task it is applied to. In subsequent chapters, this framework was applied to IDP/IDR prediction, SSP prediction and phosphorylation site prediction in **Section 6.2**, **Section 6.3** and **Section 7.5**, respectively. Since these three applications were derived from the same deep learning framework, the steps required for constructing corresponding predictive models were reduced to data collection, feature selection and hyper-parameter tuning. In practice, the DeepS2P framework can be further applied and deployed for predicting any protein sequence-base residue-level property.

In this thesis, we also proposed the multitask framework, Multi-task and Cross-stitch, that can be applied to predict any pair of correlated protein structural properties, the deep transfer framework, DeepTransfer, that can be applied to predict functionally important sites of any enzymes in post-translational modification, and the application framework of prot2vec that applies the distributed representation to any sequence-level, residue-level and biological interaction-level prediction tasks.

These deep learning frameworks represent the significant effort of developing conceptual level modelling for groups of prediction tasks and, hopefully, will facilitate more effective and efficient model design in future work.

## 9.2 Future works

The research work conducted in this thesis can be extended into multiple research areas. In this section, we highlight three of the potential directions.

### **Distributed representation of proteins based on molecule structures.**

Compared to protein sequences, protein structures are more directly related to protein functions. In particular, protein structures are partially determined by long-disorder contacts which indicates the spatial local correlation among amino acid residues [64]. Therefore, it is critical to develop the distributed representation of proteins based on molecule structures to capture the spatial local correlation. When there is sufficient data

for protein structures, the distributed representation can be extended to better represent the function of proteins.

**Quantitative correlation exploration for multiple protein structural and functional properties.** In this thesis, we applied the cross-stitch deep learning framework for predicting protein intrinsic disorder and secondary structure populations, which also generated the quantitative correlation heat map between these two structural properties. In future works, more protein structural and functional properties can be involved to produce a full correlation heat map for protein structures and functions. This heat map will be indicative for future directions of investigation in biological research. It will also provide a useful heuristic for determining which structural and functional properties should be predicted simultaneously to achieved improved performance.

**Deep learning framework design for genomic data analysis.** Proteins represent one of the most important gene products of gene expression. Their functions are primarily determined by amino acid sequences. Also, they are further translated from gene sequences. Since protein sequences and gene sequences are both biological sequences, the deep learning techniques, such as the distributed representation, the multitask deep learning frameworks and the DeepTransfer framework, which have been applied to protein analysis, can be extended and applied to genomic data analysis. It is noteworthy that the analysis of genomic data is more challenging due to the larger amount of data involved, more complex data structures and higher levels of uncertainty.

## References

1. Luo, J., et al., *Big Data Application in Biomedical Research and Health Care: A Literature Review*. Biomed Inform Insights, 2016. **8**: p. 1-10.
2. National Research Council (US) Committee for Monitoring the Nation's Changing Needs for Biomedical, B., and Clinical Personnel. *Emerging Fields and Interdisciplinary Studies*. Advancing the Nation's Health Needs: NIH Research Training Programs 2005; Available from: <https://www.ncbi.nlm.nih.gov/books/NBK22616/>.
3. Bengio, Y. and O. Delalleau. *On the expressive power of deep architectures*. in *Algorithmic Learning Theory*. 2011. Springer.
4. Phillips, T., *Regulation of transcription and gene expression in eukaryotes*. Nature Education, 2008. **1**(1): p. 199.
5. Krizhevsky, A., I. Sutskever, and G.E. Hinton. *Imagenet classification with deep convolutional neural networks*. in *Advances in neural information processing systems*. 2012.
6. Deng, L., G. Hinton, and B. Kingsbury. *New types of deep neural network learning for speech recognition and related applications: An overview*. in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. 2013. IEEE.
7. Gao, J., Y. Yang, and Y. Zhou, *Predicting the errors of predicted local backbone angles and non-local solvent- accessibilities of proteins by deep neural networks*. Bioinformatics, 2016. **32**(24): p. 3768-3773.
8. Heffernan, R., et al., *Improving prediction of secondary structure, local backbone angles, and solvent accessible surface area of proteins by iterative deep learning*. Scientific Reports, 2015. **5**: p. 11476.
9. Wang, S., et al., *Protein secondary structure prediction using deep convolutional neural fields*. Scientific reports, 2016. **6**.
10. Leung, M.K., et al., *Deep learning of the tissue-regulated splicing code*. Bioinformatics, 2014. **30**(12): p. i121-9.
11. Angermueller, C., et al., *DeepCpG: accurate prediction of single-cell DNA methylation states using deep learning*. Genome biology, 2017. **18**(1): p. 67.
12. Manning, C.D. and H. Schütze, *Foundations of statistical natural language processing*. Vol. 999. 1999: MIT Press.
13. Rabiner, L.R. and B.-H. Juang, *Fundamentals of speech recognition*. 1993.
14. Agarwal, P.K., et al., *Conformational Sub-states and Populations in Enzyme Catalysis*, in *Methods in enzymology*. 2016, Elsevier. p. 273-297.
15. Bruce, T.C., *Computational approaches: reaction trajectories, structures, and atomic motions. Enzyme reactions and proficiency*. Chemical reviews, 2006. **106**(8): p. 3119-3139.

16. Hicke, L. and R. Dunn, *Regulation of membrane protein transport by ubiquitin and ubiquitin-binding proteins*. Annual review of cell and developmental biology, 2003. **19**(1): p. 141-172.
17. Wang, L. and S.J. Brown, *BindN: a web-based tool for efficient prediction of DNA and RNA binding sites in amino acid sequences*. Nucleic acids research, 2006. **34**(suppl\_2): p. W243-W248.
18. Berg, J.M., J.L. Tymoczko, and L. Stryer, *Protein structure and function*. Biochemistry, 2002. **262**: p. 159-173.
19. Deng, L. *Achievements and Challenges of Deep Learning—From Speech Analysis and Recognition to Language and Multimodal Processing*. in *Fifteenth Annual Conference of the International Speech Communication Association*. 2014.
20. Long, M. and J. Wang, *Learning multiple tasks with deep relationship networks*. arXiv preprint arXiv:1506.02117, 2015.
21. Pereira, J.C., E.R. Caffarena, and C.N. dos Santos, *Boosting Docking-based Virtual Screening with Deep Learning*. Journal of Chemical Information and Modeling, 2016.
22. Deng, J., et al. *Imagenet: A large-scale hierarchical image database*. in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. 2009. IEEE.
23. Napoles, C., Matthew Gormley, and Benjamin Van Durme, *Annotated English Gigaword LDC2012T21*. 2012, Philadelphia: Linguistic Data Consortium.
24. Tramontano, A. and D. Cozzetto, *The Relationship Between Protein Sequence, Structure and Function*, in *Supramolecular Structure and Function 8*, G. Pifat-Mrzljak, Editor. 2004, Springer US: Boston, MA. p. 15-29.
25. Baumberg, S., *Prokaryotic gene expression*. Vol. 21. 1999: OUP Oxford.
26. Mensink, K.A. and W.E. Highsmith Jr, *Chapter 5 - Basic Concepts in Human Molecular Genetics A2 - Coleman, William B*, in *Molecular Pathology*, G.J. Tsongalis, Editor. 2009, Academic Press: San Diego. p. 89-107.
27. Griffiths AJF, M.J., Suzuki DT, *Transcription and RNA polymerase*, in *An Introduction to Genetic Analysis*. 2000.
28. Picknett, T.M. and S. Brenner, *Posttranscriptional Modification*, in *Encyclopedia of Genetics*. 2001, Academic Press: New York. p. 1532.
29. Rogers, J. and R. Wall, *A mechanism for RNA splicing*. Proceedings of the National Academy of Sciences, 1980. **77**(4): p. 1877-1879.
30. Fong, N. and D.L. Bentley, *Capping, splicing, and 3' processing are independently stimulated by RNA polymerase II: different functions for different segments of the CTD*. Genes & Development, 2001. **15**(14): p. 1783-1795.
31. *translation / RNA translation*. Available from: <https://www.nature.com/scitable/definition/translation-rna-translation-173>.
32. Liljas, A., *Translation A2 - Brenner, Sydney*, in *Encyclopedia of Genetics*, J.H. Miller, Editor. 2001, Academic Press: New York. p. 1999-2002.

33. Trost, B., et al., *DAPPLE 2: a Tool for the Homology-Based Prediction of Post-Translational Modification Sites*. J Proteome Res, 2016. **15**(8): p. 2760-7.
34. Marks, F., et al., *Protein Phosphorylation*. 2008.
35. Blom, N., S. Gammeltoft, and S. Brunak, *Sequence and structure-based prediction of eukaryotic protein phosphorylation sites*. Journal of molecular biology, 1999. **294**(5): p. 1351-1362.
36. Wright, P.E. and H.J. Dyson, *Intrinsically unstructured proteins: re-assessing the protein structure-function paradigm*. Journal of molecular biology, 1999. **293**(2): p. 321-331.
37. Anfinsen, C.B., *Principles that govern the folding of protein chains*. Science, 1973. **181**(4096): p. 223-230.
38. Le Borgne, R. and B. Hoflack, *Protein transport from the secretory to the endocytic pathway in mammalian cells*. Biochimica et Biophysica Acta (BBA)-Molecular Cell Research, 1998. **1404**(1-2): p. 195-209.
39. Consortium., U., *UniProt: a hub for protein information*. Nucleic Acids Research, 2015. **43**(D1): p. D204-D212.
40. Rose, P.W., et al., *The RCSB Protein Data Bank: views of structural biology for basic and applied research and education*. Nucleic acids research, 2015. **43**(D1): p. D345-D356.
41. Engelking, L.R., *Chapter 4 - Protein Structure*, in *Textbook of Veterinary Physiological Chemistry (Third Edition)*. 2015, Academic Press: Boston. p. 18-25.
42. Rules, I.-I.T., *A One-Letter Notation for Amino Acid Sequences\**. European Journal of Biochemistry, 1968. **5**(2): p. 151-153.
43. Dobson, C.M., *Protein folding and misfolding*. Nature, 2003. **426**(6968): p. 884-890.
44. Schermann, J.-P., *4.2 - Amino Acids, Peptides and Proteins*, in *Spectroscopy and Modeling of Biomolecular Building Blocks*. 2008, Elsevier: Amsterdam. p. 251-296.
45. David Eckersall, P., *Chapter 5 - Proteins, Proteomics, and the Dysproteinemias A2 - Kaneko, J. Jerry*, in *Clinical Biochemistry of Domestic Animals (Sixth Edition)*, J.W. Harvey and M.L. Bruss, Editors. 2008, Academic Press: San Diego. p. 117-155.
46. Ruvinsky, A.M., et al., *STRUCTURE FLUCTUATIONS AND CONFORMATIONAL CHANGES IN PROTEIN BINDING*. J Bioinform Comput Biol, 2012. **10**(2): p. 1241002.
47. Gromiha, M.M., *Chapter 6 - Protein Stability*, in *Protein Bioinformatics*. 2010, Academic Press: Singapore. p. 209-245.
48. Dunker, A.K., et al., *Intrinsically disordered protein*. Journal of Molecular Graphics and Modelling, 2001. **19**(1): p. 26-59.
49. Casem, M.L., *Chapter 3 - Proteins*, in *Case Studies in Cell Biology*. 2016, Academic Press: Boston. p. 23-71.
50. Litalien, C. and P. Beaulieu, *Chapter 117 - Molecular Mechanisms of Drug Actions: From Receptors to Effectors A2 - Fuhrman, Bradley P*, in *Pediatric Critical Care (Fourth Edition)*, J.J. Zimmerman, Editor. 2011, Mosby: Saint Louis. p. 1553-1568.

51. Weber, I.T., et al., *Chapter 3 - Reaction Intermediates Discovered in Crystal Structures of Enzymes*, in *Advances in Protein Chemistry and Structural Biology*, C. Christov and T. Karabancheva-Christova, Editors. 2012, Academic Press. p. 57-86.
52. Rarey, M., et al., *A fast flexible docking method using an incremental construction algorithm*. Journal of molecular biology, 1996. **261**(3): p. 470-489.
53. Bogosian, G., et al., *Biosynthesis and incorporation into protein of norleucine by Escherichia coli*. Journal of Biological Chemistry, 1989. **264**(1): p. 531-539.
54. Edman, P. and G. Begg, *A protein sequenator*. The FEBS Journal, 1967. **1**(1): p. 80-91.
55. Mann, M., et al., *Analysis of protein phosphorylation using mass spectrometry: deciphering the phosphoproteome*. Trends in biotechnology, 2002. **20**(6): p. 261-268.
56. Drenth, J., *Principles of protein X-ray crystallography*. 2007: Springer Science & Business Media.
57. Wüthrich, K., *NMR (Nuclear Magnetic Resonance) in biological research: peptides and proteins*. 1976.
58. Kabsch, W. and C. Sander, *Dictionary of protein secondary structure: pattern recognition of hydrogen - bonded and geometrical features*. Biopolymers, 1983. **22**(12): p. 2577-2637.
59. Zhang, J., et al., *Prediction of protein solvent accessibility using PSO-SVR with multiple sequence-derived features and weighted sliding window scheme*. BioData mining, 2015. **8**(1): p. 3.
60. Gromiha, M.M., *Chapter 3 - Protein Structure Analysis*, in *Protein Bioinformatics*. 2010, Academic Press: Singapore. p. 63-105.
61. Song, J., et al., *Prodepth: predict residue depth by support vector regression approach from protein sequences only*. PloS one, 2009. **4**(9): p. e7072.
62. Hou, G., et al., *Chapter Five - Magic Angle Spinning NMR Studies of Protein Assemblies: Recent Advances in Methodology and Applications*, in *Annual Reports on NMR Spectroscopy*, G.A. Webb, Editor. 2013, Academic Press. p. 293-357.
63. Ramachandran, G.N., C. Ramakrishnan, and V. Sasisekharan, *Stereochemistry of polypeptide chain configurations*. Journal of molecular biology, 1963. **7**(1): p. 95-99.
64. Di Lena, P., K. Nagata, and P. Baldi, *Deep architectures for protein contact map prediction*. Bioinformatics, 2012. **28**(19): p. 2449-57.
65. Uversky, V.N., C.J. Oldfield, and A.K. Dunker, *Intrinsically disordered proteins in human diseases: introducing the D2 concept*. Annu. Rev. Biophys., 2008. **37**: p. 215-246.
66. Camilloni, C., et al., *Determination of secondary structure populations in disordered states of proteins using nuclear magnetic resonance chemical shifts*. Biochemistry, 2012. **51**(11): p. 2224-2231.
67. Copley, R.R., *Protein Families: Evolution*. eLS, 2001.
68. Sigrist, C.J., et al., *PROSITE: a documented database using patterns and profiles as motif descriptors*. Briefings in bioinformatics, 2002. **3**(3): p. 265-274.

69. Finn, R.D., J. Clements, and S.R. Eddy, *HMMER web server: interactive sequence similarity searching*. Nucleic acids research, 2011: p. W29-37.
70. Finn, R.D., et al., *The Pfam protein families database: towards a more sustainable future*. Nucleic acids research, 2016. **44**(D1): p. D279-D285.
71. Das, S., et al., *CATH FunFHMmer web server: protein functional annotations using functional family assignments*. Nucleic acids research, 2015. **43**(W1): p. W148-W153.
72. Sebestyen, A., et al., *The hypervariable D3 domain of Salmonella flagellin is an autonomous folding unit*. Protein Pept Lett, 2008. **15**(1): p. 54-7.
73. Hleap, J.S., E. Susko, and C. Blouin, *Defining structural and evolutionary modules in proteins: a community detection approach to explore sub-domain architecture*. BMC Structural Biology, 2013. **13**(1): p. 20.
74. Jin, J., et al., *Eukaryotic protein domains as functional units of cellular evolution*. Science signaling, 2009. **2**(98): p. ra76-ra76.
75. DeLuca, D.C. and J. Lyndal York, *Enzymes A2 - Brenner, Sydney*, in *Encyclopedia of Genetics*, J.H. Miller, Editor. 2001, Academic Press: New York. p. 625-626.
76. Koshland, D.E., *The key - lock theory and the induced fit theory*. Angewandte Chemie International Edition, 1995. **33**(23 - 24): p. 2375-2378.
77. Xue, Y., et al., *GPS 2.1: enhanced prediction of kinase-specific phosphorylation sites with an algorithm of motif length selection*. Protein Engineering Design and Selection, 2010. **24**(3): p. 255-260.
78. Wang, D., et al., *MusiteDeep: a deep-learning framework for general and kinase-specific phosphorylation site prediction*. Bioinformatics, 2017. **33**(24): p. 3909-3916.
79. Chakrabarti, S. and C.J. Lanczycki, *Analysis and prediction of functionally important sites in proteins*. Protein Science : A Publication of the Protein Society, 2007. **16**(1): p. 4-13.
80. Amaratunga, D., H. Göhlmann, and P.J. Peeters, *3.05 - Microarrays A2 - Taylor, John B*, in *Comprehensive Medicinal Chemistry II*, D.J. Triggle, Editor. 2007, Elsevier: Oxford. p. 87-106.
81. Cheng, F.X., et al., *Prediction of Drug-Target Interactions and Drug Repositioning via Network-Based Inference*. Plos Computational Biology, 2012. **8**(5).
82. De Las Rivas, J. and C. Fontanillo, *Protein-Protein Interactions Essentials: Key Concepts to Building and Analyzing Interactome Networks*. PLoS Comput Biol, 2010. **6**(6).
83. Sangrador, A., *Protein classification: An introduction to EMBL-EBI resources*.
84. Wu, C.H., et al., *PIRSF: family classification system at the Protein Information Resource*. Nucleic acids research, 2004. **32**(suppl\_1): p. D112-D114.
85. Finn, R.D., et al., *InterPro in 2017—beyond protein family and domain annotations*. Nucleic acids research, 2016. **45**(D1): p. D190-D199.



86. Apweiler, R., et al., *The InterPro database, an integrated documentation resource for protein families, domains and functional sites*. Nucleic acids research, 2001. **29**(1): p. 37-40.
87. Eddy, S.R., *Profile hidden Markov models*. Bioinformatics, 1998. **14**(9): p. 755-763.
88. Sillitoe, I., et al., *CATH: comprehensive structural and functional annotations for genome sequences*. Nucleic acids research, 2014. **43**(D1): p. D376-D381.
89. Sickmeier, M., et al., *DisProt: the database of disordered proteins*. Nucleic acids research, 2007. **35**(suppl 1): p. D786-D793.
90. Di Domenico, T., et al., *MobiDB: a comprehensive database of intrinsic protein disorder annotations*. Bioinformatics, 2012. **28**(15): p. 2080-2081.
91. Potenza, E., et al., *MobiDB 2.0: an improved database of intrinsically disordered and mobile proteins*. Nucleic acids research, 2014. **43**(D1): p. D315-D320.
92. Piovesan, D., et al., *MobiDB 3.0: more annotations for intrinsic disorder, conformational diversity and interactions in proteins*. Nucleic acids research, 2017. **46**(D1): p. D471-D476.
93. Ulrich, E.L., et al., *BioMagResBank*. Nucleic acids research, 2007. **36**(suppl\_1): p. D402-D408.
94. Dinkel, H., et al., *Phospho. ELM: a database of phosphorylation sites—update 2011*. Nucleic acids research, 2011. **39**(suppl 1): p. D261-D267.
95. Kanehisa, M., et al., *From genomics to chemical genomics: new developments in KEGG*. Nucleic acids research, 2006. **34**(suppl 1): p. D354-D357.
96. Kanehisa, M. and S. Goto, *KEGG: kyoto encyclopedia of genes and genomes*. Nucleic acids research, 2000. **28**(1): p. 27-30.
97. Wishart, D.S., et al., *DrugBank: a knowledgebase for drugs, drug actions and drug targets*. Nucleic acids research, 2008. **36**(suppl 1): p. D901-D906.
98. Durek, P., et al., *PhosPhAt: the Arabidopsis thaliana phosphorylation site database. An update*. Nucleic acids research, 2010. **38**(suppl 1): p. D828-D834.
99. Altschul, S.F., et al., *Gapped BLAST and PSI-BLAST: a new generation of protein database search programs*. Nucleic Acids Research, 1997. **25**(17): p. 3389-3402.
100. Bhagwat, M. and L. Aravind, *PSI-blast tutorial*. Comparative Genomics, 2008: p. 177-186.
101. Jones, D.T. and D. Cozzetto, *DISOPRED3: precise disordered region predictions with annotated protein-binding activity*. Bioinformatics, 2014. **31**(6): p. 857-863.
102. Ward, J.J., et al., *The DISOPRED server for the prediction of protein disorder*. Bioinformatics, 2004. **20**(13): p. 2138-2139.
103. McGuffin, L.J., K. Bryson, and D.T. Jones, *The PSIPRED protein structure prediction server*. Bioinformatics, 2000. **16**(4): p. 404-405.
104. Buchan, D.W., et al., *Scalable web services for the PSIPRED Protein Analysis Workbench*. Nucleic acids research, 2013. **41**(W1): p. W349-W357.

105. LeCun, Y., Y. Bengio, and G. Hinton, *Deep learning*. Nature, 2015. **521**(7553): p. 436-444.
106. Jain, A.K., *Fundamentals of digital image processing*. 1989: Prentice-Hall, Inc.
107. Ivakhnenko, A.G. and V.G. Lapa, *Cybernetic predicting devices*. 1966, PURDUE UNIV LAFAYETTE IND SCHOOL OF ELECTRICAL ENGINEERING.
108. Bengio, Y., P. Simard, and P. Frasconi, *Learning long-term dependencies with gradient descent is difficult*. IEEE transactions on neural networks, 1994. **5**(2): p. 157-166.
109. Hochreiter, S., et al., *Gradient flow in recurrent nets: the difficulty of learning long-term dependencies*. 2001, A field guide to dynamical recurrent neural networks. IEEE Press.
110. Hinton, G.E., *Learning multiple layers of representation*. Trends in cognitive sciences, 2007. **11**(10): p. 428-434.
111. Bengio, Y., *Learning deep architectures for AI*. Foundations and trends® in Machine Learning, 2009. **2**(1): p. 1-127.
112. Srivastava, N., et al., *Dropout: A simple way to prevent neural networks from overfitting*. The Journal of Machine Learning Research, 2014. **15**(1): p. 1929-1958.
113. Sanders, J. and E. Kandrot, *CUDA by example: an introduction to general-purpose GPU programming*. 2010: Addison-Wesley Professional.
114. Hsu, K.-Y., H.-Y. Li, and D. Psaltis, *Holographic implementation of a fully connected neural network*. Proceedings of the IEEE, 1990. **78**(10): p. 1637-1645.
115. Hubel, D.H. and T.N. Wiesel, *Receptive fields, binocular interaction and functional architecture in the cat's visual cortex*. The Journal of physiology, 1962. **160**(1): p. 106-154.
116. LeCun, Y. and Y. Bengio, *Convolutional networks for images, speech, and time series*. The handbook of brain theory and neural networks, 1995. **3361**(10).
117. Hochreiter, S. and J. Schmidhuber, *Long short-term memory*. Neural computation, 1997. **9**(8): p. 1735-1780.
118. Mikolov, T., et al. *Recurrent neural network based language model*. in *Interspeech*. 2010.
119. Goodfellow, I., et al. *Generative adversarial nets*. in *Advances in neural information processing systems*. 2014.
120. Zhang, H., et al., *Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks*. arXiv preprint arXiv:1612.03242, 2016.
121. Salakhutdinov, R. and G.E. Hinton. *Deep boltzmann machines*. in *International conference on artificial intelligence and statistics*. 2009.
122. Hinton, G.E., S. Osindero, and Y.-W. Teh, *A fast learning algorithm for deep belief nets*. Neural computation, 2006. **18**(7): p. 1527-1554.
123. Poon, H. and P. Domingos, *Sum-Product Networks: A New Deep Architecture*. 2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops), 2011.
124. Pratt, L.Y. *Discriminability-based transfer between neural networks*. in *Advances in neural information processing systems*. 1993.

125. Caruana, R., *Multitask learning*, in *Learning to learn*. 1998, Springer. p. 95-133.
126. Zhou, W., et al., *Learning low dimensional convolutional neural networks for high-resolution remote sensing image retrieval*. Remote Sensing, 2017. **9**(5): p. 489.
127. Koller, D. and N. Friedman, *Probabilistic graphical models: principles and techniques*. 2009: MIT press.
128. Altman, N.S., *An introduction to kernel and nearest-neighbor nonparametric regression*. The American Statistician, 1992. **46**(3): p. 175-185.
129. Gunning, D., *Explainable artificial intelligence (xai)*. Defense Advanced Research Projects Agency (DARPA), nd Web, 2017.
130. Livni, R., S. Shalev-Shwartz, and O. Shamir. *On the computational efficiency of training neural networks*. in *Advances in Neural Information Processing Systems*. 2014.
131. Provost, R.K.F., *Glossary of terms*. Machine Learning, 1998. **30**.
132. Samuel, A.L., *Some Studies in Machine Learning Using the Game of Checkers. I*, in *Computer Games I*, D.N.L. Levy, Editor. 1988, Springer New York: New York, NY. p. 335-365.
133. Bishop, C.M., *Pattern recognition and machine learning*. 2006: springer.
134. Michalski, R.S., J.G. Carbonell, and T.M. Mitchell, *Machine learning: An artificial intelligence approach*. 2013: Springer Science & Business Media.
135. Mannila, H. *Data mining: machine learning, statistics, and databases*. in *Proceedings of 8th International Conference on Scientific and Statistical Data Base Management*. 1996.
136. Rasheed, S., *Data Science for Suicide Bombings: Can You Predict the Next Attack?* 2016: iUniverse.
137. Jones, D.T., *Protein secondary structure prediction based on position-specific scoring matrices*. Journal of molecular biology, 1999. **292**(2): p. 195-202.
138. Magnan, C.N. and P. Baldi, *SSpro/ACCpro 5: almost perfect prediction of protein secondary structure and relative solvent accessibility using profiles, machine learning and structural similarity*. Bioinformatics, 2014. **30**(18): p. 2592-2597.
139. Faraggi, E., et al., *SPINE X: improving protein secondary structure prediction by multistep learning coupled with prediction of solvent accessible surface area and backbone torsion angles*. Journal of computational chemistry, 2012. **33**(3): p. 259-267.
140. Wang, G.C. and C.L. Jain, *Regression analysis: modeling & forecasting*. 2003: Institute of Business Forec.
141. Song, J., et al., *TANGLE: two-level support vector regression approach for protein backbone torsion angle prediction from primary sequences*. PLoS One, 2012. **7**(2): p. e30361.
142. Li, W. and A. Godzik, *Cd-hit: a fast program for clustering and comparing large sets of protein or nucleotide sequences*. Bioinformatics, 2006. **22**(13): p. 1658-1659.
143. Zhu, X., *Semi-supervised learning literature survey*. 2005.
144. Specht, D.F., *Probabilistic neural networks*. Neural networks, 1990. **3**(1): p. 109-118.

145. Cortes, C. and V. Vapnik, *Support-vector networks*. Machine learning, 1995. **20**(3): p. 273-297.
146. Quinlan, J.R., *Induction of decision trees*. Machine learning, 1986. **1**(1): p. 81-106.
147. Muggleton, S. and L. De Raedt, *Inductive logic programming: Theory and methods*. The Journal of Logic Programming, 1994. **19**: p. 629-679.
148. Deb, K., et al., *A fast and elitist multiobjective genetic algorithm: NSGA-II*. IEEE transactions on evolutionary computation, 2002. **6**(2): p. 182-197.
149. Sutton, R.S. and A.G. Barto, *Reinforcement learning: An introduction*. Vol. 1. 1998: MIT press Cambridge.
150. Jain, A.K. and R.C. Dubes, *Algorithms for clustering data*. 1988.
151. Haykin, S. and N. Network, *A comprehensive foundation*. Neural networks, 2004. **2**(2004): p. 41.
152. Ngwar, M. and J. Wight. *A fully integrated analog neuron for dynamic multi-layer perceptron networks*. in *Neural Networks (IJCNN), 2015 International Joint Conference on*. 2015. IEEE.
153. Hanson, J., et al., *Improving protein disorder prediction by deep bidirectional long short-term memory recurrent neural networks*. Bioinformatics, 2017. **33**(5): p. 685-692.
154. Blom, N., et al., *Prediction of post - translational glycosylation and phosphorylation of proteins from the amino acid sequence*. Proteomics, 2004. **4**(6): p. 1633-1649.
155. Qian, N. and T.J. Sejnowski, *Predicting the secondary structure of globular proteins using neural network models*. Journal of molecular biology, 1988. **202**(4): p. 865-884.
156. Liu, F., C. Shen, and G. Lin. *Deep convolutional neural fields for depth estimation from a single image*. in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015.
157. Wang, S., et al., *Protein Secondary Structure Prediction Using Deep Convolutional Neural Fields*. Scientific Reports, 2016. **6**: p. 18962.
158. Wang, S., et al., *DeepCNF-D: predicting protein order/disorder regions by weighted deep convolutional neural fields*. International journal of molecular sciences, 2015. **16**(8): p. 17315-17330.
159. Elman, J.L., *Distributed representations, simple recurrent networks, and grammatical structure*. Machine learning, 1991. **7**(2-3): p. 195-225.
160. Schuster, M. and K.K. Paliwal, *Bidirectional recurrent neural networks*. IEEE Transactions on Signal Processing, 1997. **45**(11): p. 2673-2681.
161. Chung, J., et al., *Empirical evaluation of gated recurrent neural networks on sequence modeling*. arXiv preprint arXiv:1412.3555, 2014.
162. El Hhi, S. and Y. Bengio. *Hierarchical Recurrent Neural Networks for Long-Term Dependencies*. Citeseer.
163. Ganapathiraju, M., et al., *Computational biology and language*. Ambient Intelligence for Scientific Discovery, 2005: p. 25-47.

164. Li, Y., et al. *Word Embedding Revisited: A New Representation Learning and Explicit Matrix Factorization Perspective*. in *IJCAI*. 2015.
165. Salton, G. and C. Buckley, *Term-weighting approaches in automatic text retrieval*. Information processing & management, 1988. **24**(5): p. 513-523.
166. Harris, Z.S., *Distributional structure*. Word, 1954. **10**(2-3): p. 146-162.
167. Tosh, C.R. and G.D. Ruxton, *Modelling perception with artificial neural networks*. 2010: Cambridge University Press.
168. Hinton, G.E., J.L. McClelland, and D.E. Rumelhart, *Distributed representations*. Parallel distributed processing: Explorations in the microstructure of cognition, 1986. **1**(3): p. 77-109.
169. Le, Q.V. and T. Mikolov. *Distributed Representations of Sentences and Documents*. in *The International Conference on Machine Learning*. 2014.
170. Mikolov, T. and J. Dean, *Distributed representations of words and phrases and their compositionality*. 2013. **NIPS**: p. 3111-3119.
171. Mikolov, T., et al., *Efficient estimation of word representations in vector space*. arXiv preprint arXiv:1301.3781, 2013.
172. Morin, F. and Y. Bengio. *Hierarchical Probabilistic Neural Network Language Model*. in *Aistats*. 2005.
173. Gutmann, M.U. and A. Hyvärinen, *Noise-contrastive estimation of unnormalized statistical models, with applications to natural image statistics*. Journal of Machine Learning Research, 2012. **13**(Feb): p. 307-361.
174. Kulmanov, M., M.A. Khan, and R. Hoehndorf, *DeepGO: predicting protein functions from sequence and interactions using a deep ontology-aware classifier*. Bioinformatics, 2017.
175. Wang, S., et al., *Accurate de novo prediction of protein contact map by ultra-deep learning model*. PLoS computational biology, 2017. **13**(1): p. e1005324.
176. Caruna, R. *Multitask learning: A knowledge-based source of inductive bias*. in *Machine Learning: Proceedings of the Tenth International Conference*. 1993.
177. West, J., D. Ventura, and S. Warnick, *Spring research presentation: A theoretical foundation for inductive transfer*. Brigham Young University, College of Physical and Mathematical Sciences, 2007. **1**.
178. Collobert, R. and J. Weston. *A unified architecture for natural language processing: Deep neural networks with multitask learning*. in *Proceedings of the 25th international conference on Machine learning*. 2008. ACM.
179. Qi, Y., et al., *A unified multitask architecture for predicting local protein properties*. PloS one, 2012. **7**(3): p. e32235.
180. Yang, Y. and T.M. Hospedales, *Trace Norm Regularised Deep Multi-Task Learning*. arXiv preprint arXiv:1606.04038, 2016.
181. Duong, L., et al. *Low Resource Dependency Parsing: Cross-lingual Parameter Sharing in a Neural Network Parser*. in *ACL (2)*. 2015.

182. Misra, I., et al. *Cross-stitch networks for multi-task learning*. in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016.
183. Hashimoto, K., et al., *A joint many-task model: Growing a neural network for multiple NLP tasks*. arXiv preprint arXiv:1611.01587, 2016.
184. Luis, R., L.E. Sucar, and E.F. Morales, *Inductive transfer for learning Bayesian networks*. Machine learning, 2010. **79**(1-2): p. 227-255.
185. Ablavsky, V.H., C.J. Becker, and P. Fua, *Transfer learning by sharing support vectors*. 2012.
186. Goussies, N.A., S. Ubalde, and M. Mejail, *Transfer learning decision forests for gesture recognition*. The Journal of Machine Learning Research, 2014. **15**(1): p. 3667-3690.
187. Zhou, B., et al. *Learning deep features for scene recognition using places database*. in *Advances in neural information processing systems*. 2014.
188. Long, J., E. Shelhamer, and T. Darrell. *Fully convolutional models for semantic segmentation*. in *CVPR*. 2015.
189. Levi, G. and T. Hassner. *Age and gender classification using convolutional neural networks*. in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*. 2015.
190. Venugopalan, S., et al., *Translating videos to natural language using deep recurrent neural networks*. arXiv preprint arXiv:1412.4729, 2014.
191. Nguyen, A., J. Yosinski, and J. Clune. *Deep neural networks are easily fooled: High confidence predictions for unrecognizable images*. in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015.
192. Yang, L., et al. *Similarity analysis based on sparse representation for protein sequence comparison*. in *Cybernetics (CYBCONF), 2015 IEEE 2nd International Conference on*. 2015. IEEE.
193. Asgari, E. and M.R.K. Mofrad, *Continuous Distributed Representation of Biological Sequences for Deep Proteomics and Genomics*. PLoS ONE, 2015. **10**(11): p. e0141287.
194. Sormanni, P., et al., *The s2D method: simultaneous sequence-based prediction of the statistical populations of ordered and disordered regions in proteins*. Journal of molecular biology, 2015. **427**(4): p. 982-996.
195. Becker, J., F. Maes, and L. Wehenkel, *On the encoding of proteins for disordered regions prediction*. PloS one, 2013. **8**(12): p. e82252.
196. Ma, J. and S. Wang, *AcconPred: Predicting solvent accessibility and contact number simultaneously by a multitask learning framework under the conditional neural fields model*. BioMed research international, 2015. **2015**.
197. Fay, J.C. and C.-I. Wu, *Sequence divergence, functional constraint, and selection in protein evolution*. Annual review of genomics and human genetics, 2003. **4**(1): p. 213-235.

198. Marchler-Bauer, A., et al., *CDD: a conserved domain database for interactive domain family analysis*. Nucleic acids research, 2006. **35**(suppl\_1): p. D237-D240.
199. Cavalli - Sforza, L.L. and A.W. Edwards, *Phylogenetic analysis: models and estimation procedures*. Evolution, 1967. **21**(3): p. 550-570.
200. Kawashima, S. and M. Kanehisa, *AAindex: amino acid index database*. Nucleic acids research, 2000. **28**(1): p. 374-374.
201. Li, Z.-R., et al., *PROFEAT: a web server for computing structural and physicochemical features of proteins and peptides from amino acid sequence*. Nucleic Acids Research, 2006. **34**(suppl\_2): p. W32-W37.
202. van Westen, G.J., et al., *Benchmarking of protein descriptor sets in proteochemometric modeling (part 2): modeling performance of 13 amino acid descriptor sets*. Journal of cheminformatics, 2013. **5**(1): p. 42.
203. Dou, Y., et al., *L1pred: a sequence-based prediction tool for catalytic residues in enzymes with the L1-logreg classifier*. PloS one, 2012. **7**(4): p. e35666.
204. Galperin, M.Y. and D. Frishman, *11 - Towards Automated Prediction of Protein Function from Microbial Genomic Sequences*, in *Methods in Microbiology*, A.G. Craig and J.D. Hoheisel, Editors. 1999, Academic Press. p. 245-263.
205. Huang, Y.-A., et al., *Sequence-based prediction of protein-protein interactions using weighted sparse representation model combined with global encoding*. BMC bioinformatics, 2016. **17**(1): p. 1.
206. Haworth, I.S. *Protein Solubility*. Chemistry Explained.
207. Wang, S., J. Ma, and J. Xu, *AUCpreD: proteome-level protein disorder prediction by AUC-maximized deep convolutional neural fields*. Bioinformatics, 2016. **32**(17): p. i672-i679.
208. Li, H., et al., *Deep learning methods for protein torsion angle prediction*. BMC bioinformatics, 2017. **18**(1): p. 417.
209. Gribskov, M., A.D. McLachlan, and D. Eisenberg, *Profile analysis: detection of distantly related proteins*. Proceedings of the National Academy of Sciences, 1987. **84**(13): p. 4355-4358.
210. Bucher, P., et al., *A flexible motif search technique based on generalized profiles*. Computers & chemistry, 1996. **20**(1): p. 3-23.
211. Ishida, T. and K. Kinoshita, *PrDOS: prediction of disordered protein regions from amino acid sequence*. Nucleic acids research, 2007. **35**(suppl\_2): p. W460-W464.
212. Zhang, T., et al., *SPINE-D: accurate prediction of short and long disordered regions by a single neural-network based method*. Journal of Biomolecular Structure and Dynamics, 2012. **29**(4): p. 799-813.
213. Eickholt, J. and J. Cheng, *DNdisorder: predicting protein disorder using boosting and deep networks*. BMC Bioinformatics, 2013. **14**: p. 88.
214. Xue, Y., et al., *GPS 2.0, a tool to predict kinase-specific phosphorylation sites in hierarchy*. Molecular & cellular proteomics, 2008. **7**(9): p. 1598-1608.

215. Zhou, F.-F., et al., *GPS: a novel group-based phosphorylation predicting and scoring method*. Biochemical and biophysical research communications, 2004. **325**(4): p. 1443-1448.
216. Gao, J., et al., *Musite, a tool for global prediction of general and kinase-specific phosphorylation sites*. Molecular & Cellular Proteomics, 2010. **9**(12): p. 2586-2600.
217. Iakoucheva, L.M., et al., *The importance of intrinsic disorder for protein phosphorylation*. Nucleic acids research, 2004. **32**(3): p. 1037-1049.
218. Huang, H.-D., et al., *KinasePhos: a web tool for identifying protein kinase-specific phosphorylation sites*. Nucleic acids research, 2005. **33**(suppl 2): p. W226-W229.
219. Wong, Y.-H., et al., *KinasePhos 2.0: a web server for identifying protein kinase-specific phosphorylation sites based on sequences and coupling patterns*. Nucleic acids research, 2007. **35**(suppl\_2): p. W588-W594.
220. Patrick, R., et al., *PhosphoPICK: modelling cellular context to map kinase-substrate phosphorylation events*. Bioinformatics, 2014. **31**(3): p. 382-389.
221. Dou, Y., B. Yao, and C. Zhang, *PhosphoSVM: prediction of phosphorylation sites by integrating various protein sequence attributes with a support vector machine*. Amino Acids, 2014. **46**(6): p. 1459-1469.
222. Kreegipuu, A., N. Blom, and S. Brunak, *PhosphoBase, a database of phosphorylation sites: release 2.0*. Nucleic Acids Research, 1999. **27**(1): p. 237-239.
223. Lee, T.-Y., et al., *RegPhos: a system to explore the protein kinase–substrate phosphorylation network in humans*. Nucleic acids research, 2010. **39**(suppl\_1): p. D777-D787.
224. Liu, Y. and X. Yao, *Ensemble learning via negative correlation*. Neural networks, 1999. **12**(10): p. 1399-1404.
225. Bodenmiller, B., et al., *PhosphoPep—a database of protein phosphorylation sites in model organisms*. Nature biotechnology, 2008. **26**(12): p. 1339-1340.
226. Friedman, N., D. Geiger, and M. Goldszmidt, *Bayesian network classifiers*. Machine learning, 1997. **29**(2-3): p. 131-163.
227. Keshava Prasad, T., et al., *Human protein reference database—2009 update*. Nucleic acids research, 2008. **37**(suppl\_1): p. D767-D772.
228. Stark, C., et al., *BioGRID: a general repository for interaction datasets*. Nucleic acids research, 2006. **34**(suppl 1): p. D535-D539.
229. Miller, M.L., et al., *Quantitative phosphoproteomics reveals widespread full phosphorylation site occupancy during mitosis*. Science Signaling, 2010. **3**(104).
230. Kuhn, M., et al., *Systematic identification of proteins that elicit drug side effects*. Molecular Systems Biology, 2013. **9**.
231. Bleakley, K. and Y. Yamanishi, *Supervised prediction of drug–target interactions using bipartite local models*. Bioinformatics, 2009. **25**(18): p. 2397-2403.
232. Cheng, A.C., et al., *Structure-based maximal affinity model predicts small-molecule druggability*. Nature biotechnology, 2007. **25**(1): p. 71-75.



233. Gönen, M., *Predicting drug–target interactions from chemical and genomic kernels using Bayesian matrix factorization*. Bioinformatics, 2012. **28**(18): p. 2304-2310.
234. Fakhraei, S., et al., *Network-based drug-target interaction prediction with probabilistic soft logic*. Computational Biology and Bioinformatics, IEEE/ACM Transactions on, 2014. **11**(5): p. 775-787.
235. Bleakley, K. and Y. Yamanishi, *Supervised prediction of drug-target interactions using bipartite local models*. Bioinformatics, 2009. **25**(18): p. 2397-403.
236. van Laarhoven, T., S.B. Nabuurs, and E. Marchiori, *Gaussian interaction profile kernels for predicting drug–target interaction*. Bioinformatics, 2011. **27**(21): p. 3036-3043.
237. van Laarhoven, T. and E. Marchiori, *Predicting Drug-Target Interactions for New Drug Compounds Using a Weighted Nearest Neighbor Profile*. PLoS ONE, 2013. **8**(6): p. e66952.
238. Mei, J.P., et al., *Drug-target interaction prediction by learning from local information and neighbors*. Bioinformatics, 2013. **29**(2): p. 238-45.
239. Yamanishi, Y. *Supervised bipartite graph inference*. in *Advances in Neural Information Processing Systems*. 2009.
240. Yamanishi, Y., et al., *Prediction of drug–target interaction networks from the integration of chemical and genomic spaces*. Bioinformatics, 2008. **24**(13): p. i232-i240.
241. Smith, T.F. and M.S. Waterman, *Identification of common molecular subsequences*. Journal of molecular biology, 1981. **147**(1): p. 195-197.
242. Hattori, M., et al., *SIMCOMP/SUBCOMP: chemical structure search servers for network analyses*. Nucleic acids research, 2010. **38**(suppl\_2): p. W652-W656.
243. Butina, D., *Unsupervised data base clustering based on daylight's fingerprint and Tanimoto similarity: A fast and automated way to cluster small and large data sets*. Journal of Chemical Information and Computer Sciences, 1999. **39**(4): p. 747-750.
244. Nisius, B. and J. Bajorath, *Reduction and recombination of fingerprints of different design increase compound recall and the structural diversity of hits*. Chemical biology & drug design, 2010. **75**(2): p. 152-160.
245. Spectrophores™ — *Open Babel v2.3.1 documentation*. Spectrophore is a registered trademark of Silicos NV.].
246. Perlman, L., et al., *Combining drug and gene similarity measures for drug-target elucidation*. Journal of computational biology, 2011. **18**(2): p. 133-145.
247. Kuhn, M., et al., *The SIDER database of drugs and side effects*. Nucleic acids research, 2015: p. gkv1075.
248. Organization, W.H., *The anatomical therapeutic chemical classification system with defined daily doses (ATC/DDD)*. Norway: WHO, 2006.
249. Ashburner, M., et al., *Gene Ontology: tool for the unification of biology*. Nature genetics, 2000. **25**(1): p. 25-29.
250. Tipton, K., *Nomenclature Committee of the International Union of Biochemistry and Molecular Biology (NC-IUBMB)*. Enzyme nomenclature. Recommendations 1992.

- Supplement: corrections and additions*. European journal of biochemistry/FEBS, 1994. **223**(1): p. 1.
251. Resnik, P., *Semantic similarity in a taxonomy: An information-based measure and its application to problems of ambiguity in natural language*. J. Artif. Intell. Res.(JAIR), 1999. **11**: p. 95-130.
  252. Resnik, P. *Using information content to evaluate semantic similarity in a taxonomy*. in *In Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI-95*. 1995.
  253. Wang, Y. and J. Zeng, *Predicting drug-target interactions using restricted Boltzmann machines*. Bioinformatics, 2013. **29**(13): p. i126-i134.
  254. Jacob, L. and J.-P. Vert, *Protein-ligand interaction prediction: an improved chemogenomics approach*. Bioinformatics, 2008. **24**(19): p. 2149-2156.
  255. Palma, G., M.-E. Vidal, and L. Raschid, *Drug-Target Interaction Prediction Using Semantic Similarity and Edge Partitioning*, in *The Semantic Web – ISWC 2014*, P. Mika, et al., Editors. 2014, Springer International Publishing. p. 131-146.
  256. Smullyan, R.M., *First-order logic*. 1995: Courier Corporation.
  257. Garland, T., *The scientific method as an ongoing process*. Retrieved February, 2016. **12**.
  258. Domingos, P., *A few useful things to know about machine learning*. Communications of the ACM, 2012. **55**(10): p. 78-87.
  259. Bradley, A.P., *The use of the area under the ROC curve in the evaluation of machine learning algorithms*. Pattern recognition, 1997. **30**(7): p. 1145-1159.
  260. Liu, B., et al., *Pse-in-One: a web server for generating various modes of pseudo components of DNA, RNA, and protein sequences*. Nucleic acids research, 2015. **43**(W1): p. W65-W71.
  261. King, A.J. and S.W. Wallace, *Modeling with stochastic programming*. 2012: Springer Science & Business Media.
  262. Wang, Y. and Q. Chen. *Three-Label Outdoor Scene Understanding Based on Convolutional Neural Networks*. in *International Conference on Intelligent Robotics and Applications*. 2015. Springer.
  263. Gandhi, V., *Chapter 2 - Interfacing Brain and Machine*, in *Brain-Computer Interfacing for Assistive Robotics*. 2015, Academic Press: San Diego. p. 7-63.
  264. He, Z., *4 - Phosphorylation site prediction*, in *Data Mining for Bioinformatics Applications*. 2015, Woodhead Publishing. p. 29-37.
  265. Wold, S., K. Esbensen, and P. Geladi, *Principal component analysis*. Chemometrics and intelligent laboratory systems, 1987. **2**(1-3): p. 37-52.
  266. Maaten, L.v.d. and G. Hinton, *Visualizing data using t-SNE*. Journal of machine learning research, 2008. **9**(Nov): p. 2579-2605.

267. Gagné, F., *Chapter 12 - Descriptive Statistics and Analysis in Biochemical Ecotoxicology*, in *Biochemical Ecotoxicology*. 2014, Academic Press: Oxford. p. 209-229.
268. Orum, A.M., *Case Study: Logic A2 - Smelser, Neil J*, in *International Encyclopedia of the Social & Behavioral Sciences*, P.B. Baltes, Editor. 2001, Pergamon: Oxford. p. 1509-1513.
269. Armano, G., *A direct measure of discriminant and characteristic capability for classifier building and assessment*. Information Sciences, 2015. **325**: p. 466-483.
270. Monastyrskyy, B., et al., *Evaluation of disorder predictions in CASP9*. Proteins: Structure, Function, and Bioinformatics, 2011. **79**(S10): p. 107-118.
271. Monastyrskyy, B., et al., *Assessment of protein disorder region predictions in CASP10*. Proteins: Structure, Function, and Bioinformatics, 2014. **82**(S2): p. 127-137.
272. Brodersen, K.H., et al. *The balanced accuracy and its posterior distribution*. in *Pattern recognition (ICPR), 2010 20th international conference on*. 2010. IEEE.
273. Sanger, F., *The arrangement of amino acids in proteins*. Advances in protein chemistry, 1952. **7**: p. 1-67.
274. Gamow, G., and YEas, M., in *Symposium on Information Theory in Biology*. . 1958.
275. Lipman, D.J. and W.R. Pearson, *Rapid and sensitive protein similarity searches*. Science, 1985. **227**(4693): p. 1435-1441.
276. Ganapathiraju, M., et al. *Comparative n-gram analysis of whole-genome protein sequences*. in *Proceedings of the second international conference on Human Language Technology Research*. 2002. Morgan Kaufmann Publishers Inc.
277. Chopra, S., M. Auli, and A.M. Rush. *Abstractive sentence summarization with attentive recurrent neural networks*. in *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 2016.
278. Socher, R., J. Bauer, and C.D. Manning. *Parsing with compositional vector grammars*. in *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 2013.
279. Maas, A.L., et al. *Learning word vectors for sentiment analysis*. in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*. 2011. Association for Computational Linguistics.
280. Bengio, Y., et al., *A neural probabilistic language model*. Journal of machine learning research, 2003. **3**(Feb): p. 1137-1155.
281. Levy, O. and Y. Goldberg. *Neural word embedding as implicit matrix factorization*. in *Advances in neural information processing systems*. 2014.
282. Brown, P.F., et al., *Class-based n-gram models of natural language*. Computational linguistics, 1992. **18**(4): p. 467-479.
283. Cavnar, W.B. and J.M. Trenkle, *N-gram-based text categorization*. Ann Arbor MI, 1994. **48113**(2): p. 161-175.

284. Kešelj, V., et al. *N-gram-based author profiles for authorship attribution*. in *Proceedings of the conference pacific association for computational linguistics, PACLING*. 2003.
285. Suzuki, M., N. Yamagishi, and Y.-C. Tsai. *Chinese text categorization using the character N-gram*. in *Information Theory and its Applications (ISITA), 2012 International Symposium on*. 2012. IEEE.
286. Amanchy, R., et al., *A curated compendium of phosphorylation motifs*. *Nature biotechnology*, 2007. **25**(3): p. 285-286.
287. Li, J., A. Mahajan, and M.-D. Tsai, *Ankyrin repeat: a unique motif mediating protein–protein interactions*. *Biochemistry*, 2006. **45**(51): p. 15168-15178.
288. Bairoch, A. and J.-M. Claverie, *Sequence patterns in protein kinases*. *Nature*, 1988. **331**: p. 22.
289. Mikolov, T., et al. *Distributed representations of words and phrases and their compositionality*. in *NIPS*. 2013.
290. Armstrong, J., *Cosine similarity: the similarity of two weighted vectors*. *Programming Erlang (Second edition)*. Raleigh: The Pragmatic Bookshelf, 2013.
291. Yan, R., et al., *A comparative assessment and analysis of 20 representative sequence alignment methods for protein structure prediction*. *Scientific reports*, 2013. **3**: p. 2619.
292. Moult, J., et al., *Critical assessment of methods of protein structure prediction (CASP)—round x*. *Proteins: Structure, Function, and Bioinformatics*, 2014. **82**(S2): p. 1-6.
293. Fan, S.-C. and X.-G. Zhang, *Characterizing the microenvironment surrounding phosphorylated protein sites*. *Genomics, proteomics & bioinformatics*, 2005. **3**(4): p. 213-217.
294. Kumar, N., N.P. Damle, and D. Mohanty. *Getting phosphorylated: is it necessary to be solvent accessible?* in *Proc Indian Natn Sci Acad*. 2015.
295. Li, Y.H., et al., *SVM-Prot 2016: A Web-Server for Machine Learning Prediction of Protein Functional Families from Sequence Irrespective of Similarity*. *PloS ONE*, 2016. **11**(8).
296. Fitzkee, N.C. and G.D. Rose, *Reassessing random-coil statistics in unfolded proteins*. *Proceedings of the National Academy of Sciences of the United States of America*, 2004. **101**(34): p. 12497-12502.
297. Lins, L., A. Thomas, and R. Brasseur, *Analysis of accessible surface of residues in proteins*. *Protein Sci*, 2003. **12**(7): p. 1406-17.
298. Song, J., et al., *PROSPER: an integrated feature-based tool for predicting protease substrate cleavage sites*. *PloS one*, 2012. **7**(11): p. e50300.
299. Mansoori, E.G., M.J. Zolghadri, and S.D. Katebi, *Protein superfamily classification using fuzzy rule-based classifier*. *IEEE Transactions on Nanobioscience*, 2009. **8**(1): p. 92-99.
300. Manning, G., et al., *The protein kinase complement of the human genome*. *Science*, 2002. **298**(5600): p. 1912-1934.
301. Lo Conte, L., et al., *SCOP: a structural classification of proteins database*. *Nucleic acids research*, 2000. **28**(1): p. 257-259.

302. Flower, D.R., *The lipocalin protein family: structure and function*. Biochemical Journal, 1996. **318**(1): p. 1-14.
303. Vogel, C., et al., *Structure, function and evolution of multidomain proteins*. Curr Opin Struct Biol, 2004. **14**(2): p. 208-16.
304. Babushok, D., E. Ostertag, and H. Kazazian, *Current topics in genome evolution: molecular mechanisms of new gene formation*. Cellular and molecular life sciences, 2007. **64**(5): p. 542-554.
305. Pearson, K., *LIII. On lines and planes of closest fit to systems of points in space*. The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science, 1901. **2**(11): p. 559-572.
306. Bellefroid, E., et al., *The evolutionarily conserved Kruppel-associated box domain defines a subfamily of eukaryotic multifingered proteins*. Proceedings of the National Academy of Sciences of the United States of America, 1991. **88**(9): p. 3608-3612.
307. Mikolov, T. and J. Dean, *Distributed representations of words and phrases and their compositionality*. Advances in neural information processing systems, 2013.
308. Asgari, E. and M.R.K. Mofrad, *Continuous Distributed Representation of Biological Sequences for Deep Proteomics and Genomics*. PLoS ONE, 2015. **10**(11).
309. Kerner, M.J., et al., *Proteome-wide analysis of chaperonin-dependent protein folding in Escherichia coli*. Cell, 2005. **122**(2): p. 209-220.
310. VanBogelen, R.A., et al., *Escherichia coli proteome analysis using the gene - protein database*. Electrophoresis, 1997. **18**(8): p. 1243-1251.
311. Taniguchi, Y., et al., *Quantifying E. coli proteome and transcriptome with single-molecule sensitivity in single cells*. science, 2010. **329**(5991): p. 533-538.
312. Boc, A., A.B. Diallo, and V. Makarenkov, *T-REX: a web server for inferring, validating and visualizing phylogenetic trees and networks*. Nucleic acids research, 2012. **40**(W1): p. W573-W579.
313. Marchler-Bauer, A., et al., *CDD: NCBI's conserved domain database*. Nucleic acids research, 2014: p. D222-226.
314. Paul, M.K. and A.K. Mukhopadhyay, *Tyrosine kinase – Role and significance in Cancer*. International Journal of Medical Sciences, 2004. **1**(2): p. 101-115.
315. Uhlén, M., et al., *A human protein atlas for normal and cancer tissues based on antibody proteomics*. Molecular & cellular proteomics, 2005. **4**(12): p. 1920-1932.
316. Ang, M.K., et al., *Molecular classification of breast phyllodes tumors: validation of the histologic grading scheme and insights into malignant progression*. Breast cancer research and treatment, 2011. **129**(2): p. 319-329.
317. Maruyama, I.N., *Mechanisms of Activation of Receptor Tyrosine Kinases: Monomers or Dimers*. Cells, 2014. **3**(2): p. 304-330.
318. Lemmon, M.A. and J. Schlessinger, *Cell signaling by receptor-tyrosine kinases*. Cell, 2010. **141**(7): p. 1117-1134.

319. Guo, A., et al., *Signaling networks assembled by oncogenic EGFR and c-Met*. Proceedings of the National Academy of Sciences, 2008. **105**(2): p. 692-697.
320. Schlessinger, J., *Cell signaling by receptor tyrosine kinases*. Cell, 2000. **103**(2): p. 211-225.
321. Doughman, R., A. Firestone, and R. Anderson, *Phosphatidylinositol phosphate kinases put PI4, 5P 2 in its place*. The Journal of membrane biology, 2003. **194**(2): p. 77-89.
322. Dyson, H.J. and P.E. Wright, *Intrinsically unstructured proteins and their functions*. Nature reviews Molecular cell biology, 2005. **6**(3): p. 197-208.
323. Jensen, M.R., R.W. Ruigrok, and M. Blackledge, *Describing intrinsically disordered proteins at atomic resolution by NMR*. Current opinion in structural biology, 2013. **23**(3): p. 426-435.
324. Hirose, S., et al., *POODLE-L: a two-level SVM prediction system for reliably predicting long disordered regions*. Bioinformatics, 2007. **23**(16): p. 2046-2053.
325. Kryshtafovych, A., K. Fidelis, and A. Tramontano, *Evaluation of model quality predictions in CASP9*. Proteins: Structure, Function, and Bioinformatics, 2011. **79**(S10): p. 91-106.
326. Deng, X., J. Eickholt, and J. Cheng, *PreDisorder: ab initio sequence-based prediction of protein disordered regions*. BMC bioinformatics, 2009. **10**(1): p. 436.
327. Mizianty, M.J., Z. Peng, and L. Kurgan, *MFDp2: Accurate predictor of disorder in proteins by fusion of disorder probabilities, content and profiles*. Intrinsically Disordered Proteins, 2013. **1**(1): p. e24428.
328. Peng, K., et al., *Length-dependent prediction of protein intrinsic disorder*. BMC bioinformatics, 2006. **7**(1): p. 1.
329. Wells, M., et al., *Structure of tumor suppressor p53 and its intrinsically disordered N-terminal transactivation domain*. PNAS, 2008. **105**(15).
330. Kannan, S., D. Lane, and C. Verma, *Long range recognition and selection in IDPs: the interactions of the C-terminus of p53*. Scientific reports, 2016. **6**: p. 23750.
331. Kussie, P.H., et al., *Structure of the MDM2 oncoprotein bound to the p53 tumor suppressor transactivation domain*. Science, 1996. **274**(5289): p. 948.
332. Rowell, J., et al., *HMGB1-facilitated p53 DNA binding occurs via HMG-Box/p53 transactivation domain interaction, regulated by the acidic tail*. Structure (London, England: 1993), 2012. **20**(12): p. 2014-2024.
333. Joerger, A.C., M.D. Allen, and A.R. Fersht, *Crystal Structure of a Superstable Mutant of Human p53 Core Domain Insights Into The Mechanism Of Rescuing Oncogenic Mutations*. Journal of Biological Chemistry, 2004. **279**(2): p. 1291-1296.
334. Ren, W., et al., *Structural basis of SOSS1 complex assembly and recognition of ssDNA*. Cell reports, 2014. **6**(6): p. 982-991.
335. Ubersax, J.A. and J.E. Ferrell Jr, *Mechanisms of specificity in protein phosphorylation*. Nature reviews Molecular cell biology, 2007. **8**(7): p. 530-541.

336. Pearlman, S.M., Z. Serber, and J.E. Ferrell, *A mechanism for the evolution of phosphorylation sites*. Cell, 2011. **147**(4): p. 934-946.
337. Vlastaridis, P., et al., *Estimating the total number of phosphoproteins and phosphorylation sites in eukaryotic proteomes*. GigaScience, 2017. **6**(2): p. 1-11.
338. Scientific, T.F. *Phosphorylation*. Protein Biology Resource Library 2018 [cited 2018].
339. FARRELL, P.J., T. HUNT, and R.J. JACKSON, *Analysis of Phosphorylation of Protein Synthesis Initiation Factor eIF - 2 by Two - Dimensional Gel Electrophoresis*. The FEBS Journal, 1978. **89**(2): p. 517-521.
340. Song, J., et al., *PhosphoPredict: A bioinformatics tool for prediction of human kinase-specific phosphorylation substrates and sites by integrating heterogeneous feature selection*. Scientific Reports, 2017. **7**.
341. Mihalek, I., I. Reš, and O. Lichtarge, *A family of evolution–entropy hybrid methods for ranking protein residues by importance*. Journal of molecular biology, 2004. **336**(5): p. 1265-1282.
342. Johansson, F. and H. Toh, *A comparative study of conservation and variation scores*. BMC Bioinformatics, 2010. **11**(1): p. 388.
343. Li, F., et al., *GlycoMine: a machine learning-based approach for predicting N-, C-and O-linked glycosylation in the human proteome*. Bioinformatics, 2015. **31**(9): p. 1411-1419.
344. Li, Y., et al., *Accurate in silico identification of species-specific acetylation sites by integrating protein sequence-derived and functional features*. Scientific reports, 2014. **4**.
345. Wang, M., et al., *Cascleave 2.0, a new approach for predicting caspase and granzyme cleavage targets*. Bioinformatics, 2013. **30**(1): p. 71-80.
346. Song, J., et al., *Cascleave: towards more accurate prediction of caspase substrate cleavage sites*. Bioinformatics, 2010. **26**(6): p. 752-760.
347. Taylor, W.R., *The classification of amino acid conservation*. Journal of theoretical Biology, 1986. **119**(2): p. 205-218.
348. Sweet, R.M. and D. Eisenberg, *Correlation of sequence hydrophobicities measures similarity in three-dimensional protein structure*. Journal of molecular biology, 1983. **171**(4): p. 479-488.
349. Zulawski, M., R. Braginets, and W.X. Schulze, *PhosPhAt goes kinases—searchable protein kinase target information in the plant phosphorylation site database PhosPhAt*. Nucleic acids research, 2013. **41**(D1): p. D1176-D1184.
350. Cherkassky, V. and F. Mulier, *Learning from Data: Concepts, Theory, and Methods (Adaptive and Learning Systems for Signal Processing, Communications and Control Series)*. 1998.
351. Hsu, C.-W., C.-C. Chang, and C.-J. Lin, *A practical guide to support vector classification*. 2003.
352. Hoang, C.D.V., G. Haffari, and T. Cohn, *Incorporating Side Information into Recurrent Neural Network Language Models*, in *HLT-NAACL*. 2016. p. 1250-1255.

353. Forcier, J., P. Bissex, and W.J. Chun, *Python web development with Django*. 2008: Addison-Wesley Professional.
354. Xue, Y., et al., *PPSP: prediction of PK-specific phosphorylation site with Bayesian decision theory*. BMC Bioinformatics, 2006. **7**: p. 163.
355. Fan, W., et al., *Prediction of protein kinase-specific phosphorylation sites in hierarchical structure using functional information and random forest*. Amino Acids, 2014. **46**(4): p. 1069-78.
356. Xia, Z., et al., *Semi-supervised drug-protein interaction prediction from heterogeneous biological spaces*. BMC systems biology, 2010. **4**(Suppl 2): p. S6.
357. Schomburg, I., et al., *BRENDA, the enzyme database: updates and major new developments*. Nucleic acids research, 2004. **32**(suppl 1): p. D431-D433.
358. Günther, S., et al., *SuperTarget and Matador: resources for exploring drug-target relationships*. Nucleic acids research, 2008. **36**(suppl 1): p. D919-D922.
359. Liu, Y., et al., *DCDB: drug combination database*. Bioinformatics, 2010. **26**(4): p. 587-588.
360. Davis, J. and M. Goadrich. *The relationship between Precision-Recall and ROC curves*. in *Proceedings of the 23rd international conference on Machine learning*. 2006. ACM.
361. Minker, J. *On indefinite databases and the closed world assumption*. in *International Conference on Automated Deduction*. 1982. Springer.
362. Kimmig, A., et al. *A short introduction to probabilistic soft logic*. in *Proceedings of the NIPS Workshop on Probabilistic Programming: Foundations and Applications*. 2012.
363. Murphy, K.P., *Dynamic bayesian networks*. Probabilistic Graphical Models, M. Jordan, 2002. **7**.



# Appendix A

**Table A.1** A summary of the nomenclature of functionally important sites.

Functional categories	Functionally important sites
Active site	All active site, catalytic site residues
Ligand binding site	<ol style="list-style-type: none"> <li>1. Nucleotide binding sites: DNA and RNA binding sites</li> <li>2. Lipid binding sites: cholesterol, glycerol, ganglioside, etc.</li> <li>3. Carbohydrate binding sites: glucose, fructose, lactose, maltose, disaccharides, trisaccharides, etc.</li> <li>4. Small organic ligand binding sites: ATP, ascorbate, benzamidine, butyramide, citrate, cyclosporine, FAD, FMN, hapten, heme, NAD, NADP, NADPH, oxalate, pterin, pyruvate, tetracycline, etc.</li> <li>5. Inorganic ligand binding sites: CO, phosphate, and sulfur binding sites</li> </ol>
Protein binding site	<ol style="list-style-type: none"> <li>1. Protein-protein interaction sites extracted from heterocomplexes</li> <li>2. Peptide binding sites</li> <li>3. Specific protein binding sites: actin, tubulin, heparin, etc.</li> </ol>
Metal binding site	All metal binding residues
Post-translational modification site	<ol style="list-style-type: none"> <li>1. Acetylation sites, catalysed by <math>\alpha</math>-tubulin acetyl-transferase (<math>\alpha</math>TAT1).</li> <li>2. Cleavage sites, catalysed by <i>proteases</i>.</li> <li>3. Glycosylation sites, catalysed by <i>oligosaccharyltransferase</i> (OTase) and others.</li> <li>4. Lipoylation sites, catalysed by lipoate protein ligase (LplA), or by a lipoyl transferase (LipB) plus a lipoic acid synthetase (LipA).</li> <li>5. Phosphorylation sites, catalysed by <i>protein kinases</i>.</li> </ol>
Miscellaneous site	Sites involved in structural changes, disulphide bonding residues, hinge regions, etc.

**Table A.2** A summary of performance evaluation measures.

Name	Abbreviation	Equation
True positive	TP	N/A
True negative	TN	N/A
False positive	FP	N/A
False negative	FN	N/A
Precision	PR	$PR = \frac{TP}{TP + FP}$
Sensitivity (Recall)	SE (RE)	$SE = \frac{TP}{TP + FN}$
Specificity	SP	$SP = \frac{TN}{TN + FP}$
Accuracy	ACC	$ACC = \frac{TP + TN}{TP + FP + TN + FN}$
Balanced accuracy	BACC	$BACC = \frac{1}{2} \left( \frac{TP}{TP + FN} + \frac{TN}{TN + FP} \right)$
F1-score	F1	$F1 = 2 \times \frac{PR \times RE}{PR + RE}$
Matthews coefficient of correlation	MCC	$MCC = \frac{TP \times TN + FP \times FN}{\sqrt{(TP + FN)(TP + FN)(FP + TN)(FP + FN)}}$
Pearson coefficient of correlation	PCC	$PCC = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^n (x_i - \bar{x})^2} \sqrt{\sum_{i=1}^n (y_i - \bar{y})^2}}$ where $x_i$ and $y_i$ are single samples in the true dataset $x$ and predicted dataset $y$ , respectively
True positive rate (Sensitivity, Recall)	TPR	$TPR = \frac{TP}{TP + FN}$
False positive rate	FPR	$FPR = \frac{FP}{FP + TN} = 1 - SP$
Receiver operating characteristic curve	ROC curve	The ROC curve plots the changes of TPR with respect to the changes of FPR.
Area under the ROC curve	AUC	The AUC score measures the areas under the ROC curve.

\* In this thesis, the above evaluation measurement scores were calculated using the scikit-learn machine learning package<sup>35</sup> in Python.

<sup>35</sup> Scikit-learn: <http://scikit-learn.org/stable/>

**Table A.3** The number of annotated phosphosites in training and independent test sets for each kinase node in hierarchical phosphorylation site prediction.

Kinase node	# Training	# Test	# Total
ST	65631	18138	83769
ST/AGC	2022	586	2608
ST/AGC/DMPK	89	36	125
ST/AGC/DMPK/ROCK	85	33	118
ST/AGC/DMPK/ROCK/ROCK1	36	18	54
ST/AGC/DMPK/ROCK/ROCK2	43	11	54
ST/AGC/GRK	244	64	308
ST/AGC/GRK/BARK	140	45	185
ST/AGC/GRK/BARK/GRK2	131	42	173
ST/AGC/GRK/GRKs	114	19	133
ST/AGC/GRK/GRKs/GRK6	58	11	69
ST/AGC/NDR	6	17	23
ST/AGC/NDR/NDR-	6	17	23
ST/AGC/NDR/NDR-/LATS1	13	6	19
ST/AGC/NDR/NDR-/LATS2	8	13	21
ST/AGC/PKA	700	197	897
ST/AGC/PKB	251	65	316
ST/AGC/PKB/PKB-	51	12	63
ST/AGC/PKB/PKB-/PDK1	52	11	63
ST/AGC/PKC	768	194	962
ST/AGC/PKC/Alpha	165	47	212
ST/AGC/PKC/Alpha/PKCA	157	39	196
ST/AGC/PKC/Alpha/PKCB	25	6	31
ST/AGC/PKC/Delta	46	17	63
ST/AGC/PKC/Delta/PKCD	51	12	63
ST/AGC/PKC/Eta	31	19	50
ST/AGC/PKC/Eta/PKCE	41	9	50
ST/AGC/PKC/Iota	34	13	47
ST/AGC/PKC/Iota/PKCZ	33	9	42
ST/AGC/PKG	65	24	89
ST/AGC/RSK	53	13	66

ST/AGC/SGK	77	28	105
ST/AGC/SGK/SGK-	62	23	85
ST/AGC/SGK/SGK-/SGK1	61	18	79
ST/Atypical	151	73	224
ST/Atypical/PIKK	147	73	220
ST/Atypical/PIKK/ATM	130	59	189
ST/Atypical/PIKK/ATR	23	33	56
ST/Atypical/PIKK/DNAPK	11	10	21
ST/CAMK	651	298	949
ST/CAMK/CAMK1	63	22	85
ST/CAMK/CAMK1/CAMK1-	25	12	37
ST/CAMK/CAMK1/CAMK1-/CAMK4	17	11	28
ST/CAMK/CAMK2	121	41	162
ST/CAMK/CAMK2/CAMK2-	44	17	61
ST/CAMK/CAMK2/CAMK2-/CAMK2A	27	13	40
ST/CAMK/CAMKL	278	120	398
ST/CAMK/CAMKL/AMPK	167	50	217
ST/CAMK/CAMKL/BRSK	12	21	33
ST/CAMK/CAMKL/BRSK/BRSK1	18	12	30
ST/CAMK/CAMKL/BRSK/BRSK2	21	5	26
ST/CAMK/CAMKL/CHK1s	7	9	16
ST/CAMK/CAMKL/CHK1s/CHK1	8	8	16
ST/CAMK/CAMKL/CHK2s	5	12	17
ST/CAMK/CAMKL/CHK2s/CHK2	9	8	17
ST/CAMK/CAMKL/LKB	36	12	48
ST/CAMK/CAMKL/LKB/LKB1	36	12	48
ST/CAMK/CAMKL/MARK	31	16	47
ST/CAMK/CAMKL/MARK/MARK1	18	9	27
ST/CAMK/CAMKL/NuaK	21	13	34
ST/CAMK/CAMKL/NuaK/NUAK1	24	10	34
ST/CAMK/CAMKL/PASKs	13	10	23
ST/CAMK/CAMKL/PASKs/PASK	14	9	23
ST/CAMK/DAPK	33	20	53
ST/CAMK/DAPK/DAPK-	32	20	52
ST/CAMK/DAPK/DAPK-/DAPK1	28	9	37

ST/CAMK/DAPK/DAPK-/DAPK3	10	6	16
ST/CAMK/MAPKAPK	70	49	119
ST/CAMK/MAPKAPK/MAPKAPKs	68	48	116
ST/CAMK/MAPKAPK/MAPKAPKs/MAPKAPK2	72	18	90
ST/CAMK/MAPKAPK/MAPKAPKs/MAPKAPK3	15	11	26
ST/CAMK/MAPKAPK/MAPKAPKs/MAPKAPK5	23	14	37
ST/CAMK/MLCK	14	4	18
ST/CAMK/PHK	51	50	101
ST/CAMK/PKD	50	11	61
ST/CAMK/RAD53	5	12	17
ST/CK1	225	74	299
ST/CK1/CK1f	208	64	272
ST/CK1/CK1f/CK1f-	23	17	40
ST/CK1/CK1f/CK1f-/CK1A	13	4	17
ST/CK1/TTBK	13	3	16
ST/CK1/TTBK/TTBK-	13	3	16
ST/CK1/TTBK/TTBK-/TTBK1	13	3	16
ST/CK1/VRK	23	20	43
ST/CK1/VRK/VRK-	23	20	43
ST/CK1/VRK/VRK-/VRK1	29	14	43
ST/CK1/VRK/VRK-/VRK2	24	8	32
ST/CMGC	880	319	1199
ST/CMGC/CDK	473	155	628
ST/CMGC/CDK/CDC2s	186	76	262
ST/CMGC/CDK/CDC2s/CDC2	18	5	23
ST/CMGC/CDK/CDC2s/CDK2	182	51	233
ST/CMGC/CDK/CDK4s	14	11	25
ST/CMGC/CDK/CDK4s/CDK4	12	10	22
ST/CMGC/CDK/CDK5s	131	70	201
ST/CMGC/CDK/CDK5s/CDK5	159	42	201
ST/CMGC/CDK/CDK7s	30	14	44
ST/CMGC/CDK/CDK7s/CDK7	37	7	44
ST/CMGC/CDK/CDK9s	39	5	44
ST/CMGC/CDK/CDK9s/CDK9	39	5	44
ST/CMGC/DYRK	108	73	181

ST/CMGC/DYRK/Dyrk1	17	4	21
ST/CMGC/DYRK/Dyrk2	58	42	100
ST/CMGC/DYRK/Dyrk2/DYRK2	70	26	96
ST/CMGC/DYRK/HIPK	39	38	77
ST/CMGC/DYRK/HIPK/HIPK2	49	13	62
ST/CMGC/GSK	185	86	271
ST/CMGC/GSK/GSK3A	25	9	34
ST/CMGC/GSK/GSK3B	212	59	271
ST/CMGC/MAPK	161	39	200
ST/Other	873	258	1131
ST/Other/CK2	523	145	668
ST/Other/CK2/CK2-	92	30	122
ST/Other/CK2/CK2-/CK2A	100	22	122
ST/Other/IKK	95	37	132
ST/Other/IKK/IKK-	68	37	105
ST/Other/IKK/IKK-/IKKA	17	7	24
ST/Other/IKK/IKK-/IKKB	38	14	52
ST/Other/IKK/IKK-/IKKE	18	7	25
ST/Other/IKK/IKK-/TBK1	25	10	35
ST/Other/NEK	34	10	44
ST/Other/PEK	16	5	21
ST/Other/PLK	199	62	261
ST/Other/PLK/PLK-	200	61	261
ST/Other/PLK/PLK-/PLK1	108	30	138
ST/Other/PLK/PLK-/PLK2	38	15	53
ST/Other/PLK/PLK-/PLK3	51	17	68
ST/STE	168	73	241
ST/STE/STE11	20	8	28
ST/STE/STE20	113	35	148
ST/STE/STE20/PAKA	88	35	123
ST/STE/STE20/PAKA/PAK1	45	14	59
ST/STE/STE20/PAKA/PAK2	52	14	66
ST/STE/STE20/PAKA/PAK3	23	8	31
ST/STE/STE20/PAKB	12	5	17
ST/STE/STE7	35	31	66

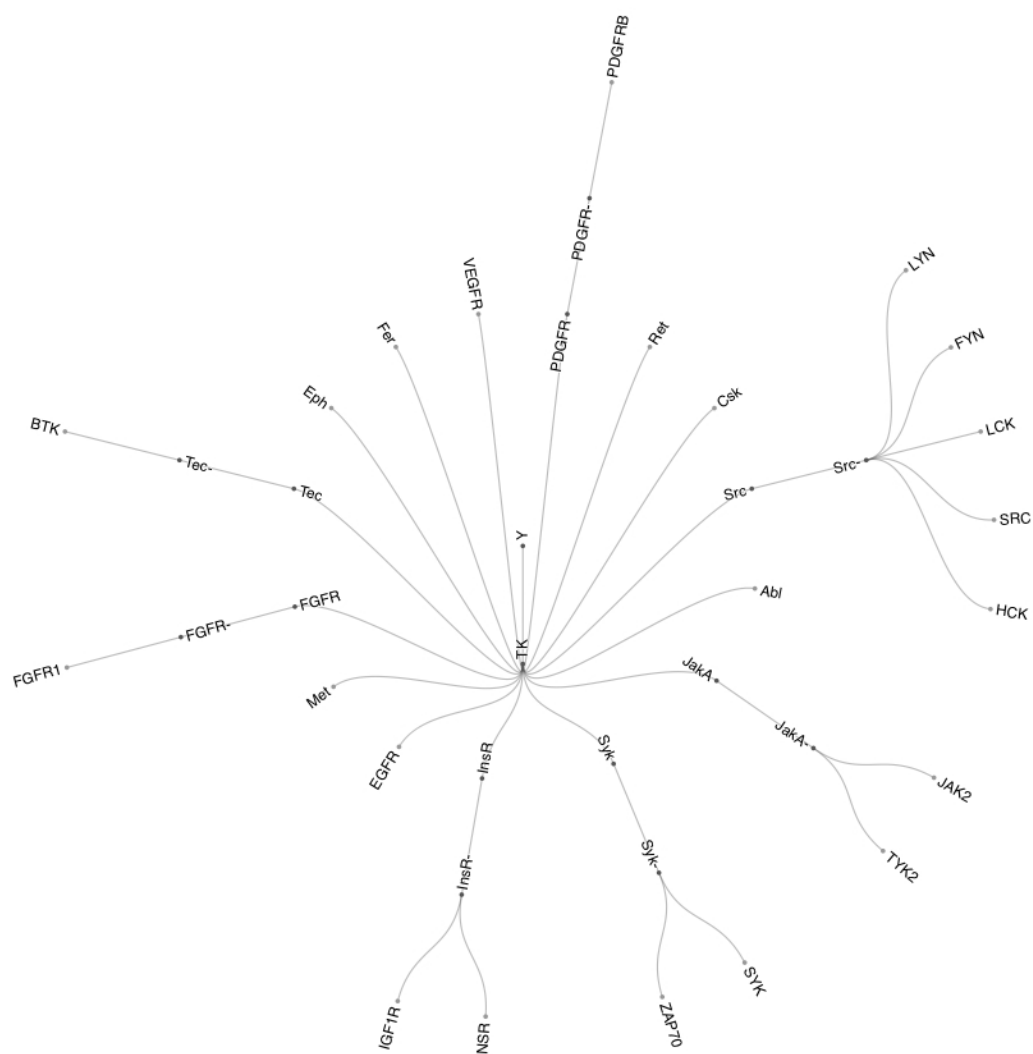
ST/STE/STE7/STE7-	37	29	66
ST/STE/STE7/STE7-/MAP2K3	21	5	26
ST/STE/STE7/STE7-/MAP2K4	18	11	29
ST/STE/STE7/STE7-/MAP2K6	20	7	27
ST/STE/STE7/STE7-/MAP2K7	16	7	23
Y	5524	1724	7248
Y/TK	966	398	1364
Y/TK/Abl	58	18	76
Y/TK/Csk	112	45	157
Y/TK/EGFR	40	48	88
Y/TK/Eph	18	4	22
Y/TK/Fer	11	12	23
Y/TK/FGFR	31	14	45
Y/TK/FGFR/FGFR-	30	14	44
Y/TK/FGFR/FGFR-/FGFR1	28	7	35
Y/TK/InsR	58	42	100
Y/TK/InsR/InsR-	58	42	100
Y/TK/InsR/InsR-/IGF1R	19	7	26
Y/TK/InsR/InsR-/INSR	65	16	81
Y/TK/JakA	72	24	96
Y/TK/JakA/JakA-	72	24	96
Y/TK/JakA/JakA-/JAK2	58	13	71
Y/TK/JakA/JakA-/TYK2	12	7	19
Y/TK/Met	4	26	30
Y/TK/PDGFR	32	26	58
Y/TK/PDGFR/PDGFR-	13	19	32
Y/TK/PDGFR/PDGFR-/PDGFRB	4	13	17
Y/TK/Ret	14	4	18
Y/TK/Src	445	186	631
Y/TK/Src/Src-	448	177	625
Y/TK/Src/Src-/FYN	79	32	111
Y/TK/Src/Src-/HCK	19	7	26
Y/TK/Src/Src-/LCK	53	21	74
Y/TK/Src/Src-/LYN	70	11	81
Y/TK/Src/Src-/SRC	298	83	381

Y/TK/Syk	66	28	94
Y/TK/Syk/Syk-	66	28	94
Y/TK/Syk/Syk-/SYK	62	17	79
Y/TK/Syk/Syk-/ZAP70	10	11	21
Y/TK/Tec	32	19	51
Y/TK/Tec/Tec-	32	19	51
Y/TK/Tec/Tec-/BTK	21	16	37
Y/TK/VEGFR	15	3	18

---







**Figure A.2** The hierarchical kinase classification tree of the constructed benchmark datasets for Tyrosine sites.

# GLOSSARY

## Bioinformatics

Anatomical Therapeutic Chemical annotations (ATC) .....	64
Average cumulative hydrophobicity (ACH).....	219
Biological interaction-level predictions .....	50
Biological word .....	49
Enzyme Commission number (E.C. number).....	65
Evolutionary homology profile .....	45
FASTA format.....	87
Gene Ontology (GO) .....	65
General phosphorylation site prediction .....	216
Hidden Markov model profile (HMM profile).....	55
Hierarchical kinase classification tree (HKCT) .....	259
Kinase-specific phosphorylation site prediction.....	216
Phylogenetic analysis.....	47
Physicochemical properties.....	46
Position-specific scoring matrix (PSSM) .....	17
Protein descriptors .....	47
Protein family classification .....	136
Protein family prediction .....	137
Protein sequence based predictions .....	49
Residue-level predictions .....	49
Sequence alignment.....	46
Sequence motif .....	48
Taylor's overlapping property (OP) .....	219
Whole sequence-level predictions.....	49

## Biology

<i>Active site</i> .....	11
Adenosine diphosphate (ADP).....	214
Adenosine triphosphate (ATP) .....	214
Amino acid.....	87
Amino acid residue (Residue) .....	6

Arabidopsis thaliana (A. thaliana).....	157
Backbone torsion angles .....	9
Biological target.....	7
Drug-target interaction (DTI) .....	12
Enzyme .....	7
Escherichia coli (E. coli) .....	155
Functionally important sites (FIS).....	12
Gene expression .....	3
Message RNA (mRNA) .....	3
NMR chemical shifts.....	17
Nuclear magnetic resonance (NMR) .....	7
Phosphoryl group .....	214
<i>Phosphorylation</i> .....	12
Phosphorylation site (phosphosite) .....	215
Phosphorylation sites .....	12
Post-transcriptional modification .....	4
Post-translational modification (PTM) .....	4
Post-translational modification site (PTM site) .....	128
Protein expression.....	6
<i>Protein family</i> .....	11
Protein functional domains (protein domains, conserved domains).....	11
Protein intrinsic disorder (IDP/IDR) .....	10
Protein residue-residue contact maps (Protein contact map) .....	10
Protein secondary structure populations (SSP) .....	10
Protein secondary structures (SS) .....	6, 9
Protein sequences.....	6
Protein structural/functional property.....	8
Protein substrate .....	215
Protein tertiary structure .....	6
<i>Protein-protein interaction (PPI)</i> .....	12
<i>Solvent accessibility</i> .....	9
The Anfinsen's dogma.....	5
Transcription.....	4
Transfer RNA (tRNA) .....	4
Translation.....	4
X-ray diffraction (X-ray) .....	7
Computer Science	

Artificial intelligence.....	29
Data mining .....	29
Machine learning .....	29
Pattern recognition .....	29
Statistics .....	29
Deep learning	
1-of-K encoding .....	38
Activation function .....	116
Alternate training .....	126
Boltzmann machines .....	19
Continuous bag-of-words (CBOW) architecture.....	39
Convolutional filters.....	34
Convolutional neural fields (CNF) .....	35
Convolutional neural network (CNN).....	18
Cross-stitch multitask learning .....	43
Data representation.....	21
Deep belief nets .....	19
Deep learning .....	17
Distributed bag-of-words architecture (DBOW).....	41
Distributed memory architecture (DM) .....	41
Distributed representation .....	39
Drop-out .....	117
Feedforward neural network language model (NNLM) .....	39
Fully-connected layer .....	117
Fully-connected neural networks.....	18
Generative adversarial network (GAN) .....	19
Hard parameter sharing .....	43
Hyper-parameter .....	118
localist representation .....	38
Max-pooling.....	116
Multitask learning .....	42
Negative constructive sampling (NCE).....	41
Overfitting .....	131
Padding strategy .....	34
Receptive fields .....	33
Recurrent neural network (RNN).....	19
SAME padding .....	34

Skip-gram architecture .....	39
Soft parameter sharing.....	43
Sum-product networks .....	19
The bidirectional RNN .....	37
The Elman network .....	37
The gated RNN .....	37
The hierarchical RNN .....	37
The long-short term memory (LSTM).....	37
Transfer learning .....	20
Underfitting .....	131
Word embedding .....	38
ZERO padding.....	34
Machine learning	
Artificial neural networks (Neural networks).....	31
Bayesian networks .....	43
Case study .....	82
Classification .....	30
Clustering .....	30
Conditional random fields (CRF) .....	35
Correlation analysis.....	82
Cross entropy (CE).....	118
Cross-validation test.....	80
Data collection.....	76
Decision forest.....	44
Decision trees.....	31
Feature construction.....	113
Feature extraction .....	113
Feature selection.....	110
Genetic algorithm .....	31
Gradient descent.....	79
Independent test.....	81
Inductive logic programming .....	31
Learning rate .....	97
Leave-one-out cross-validation (LOOCV) .....	286
Linear regression (LR).....	196
Mean squared error (MSE).....	117
Model training.....	79

Negative sampling.....	138
n-gram .....	89
Non-linear deep models .....	196
Non-linear shallow models .....	196
Objective function.....	79
Principal component analysis (PCA).....	81
Probabilistic graphical models.....	31
Regression .....	30
Reinforcement learning .....	31
Sigmoid function.....	117
Softmax function.....	118
Supervised learning.....	31, 76
Support vector machine .....	44
Support vector machine (SVM) .....	31
t-distributed stochastic neighbour embedding (t-SNE).....	81
Term-frequency-inverse document frequency (TF-IDF).....	38
Testing dataset.....	77
Training dataset.....	77
Unsupervised learning.....	31, 76